Notes on Interfaces
or Tabs to Grab
2009-

Hello,

Just a note on this “verbatim .pdf” version.

The original was done between the years of 2009 and 2011, on a PC-based word processor. Many of the examples relate more to desktop-optimized applications than to those on cell phones.

However - even though CPU speeds have gotten faster, and interfaces somewhat less clunky (well, that’s the hope, anyway) - many of the UI issues addressed here stand, even now.

There has been one instance, at least, where a plea made in this document found its fruition in an actual UI feature: The propounding of “cliche buttons”, on pages 19 - 20.

This was an attempt to ask for a handy form of clickable “cliches” or “clips”, which, in effect, would have the same effect as a clipboard - only with multiple saved items at the click of a virtual button, not only a single one.

Not exactly clear *when*, but at some point along the road, the Android OS made available a *multi-entry clipboard* feature which solves this gripe. It is standard with all current versions of Android. While it’s not identified as a “new feature” but rather simply as Android’s answer to Apple’s version of a clipboard, it *does* in fact fulfill most of what was asked for here as “cliche buttons”, and does so with some efficiency.

Let’s hope we can count on a few more such developments, as well as a little Windows-based *kaizen*, in the near future.

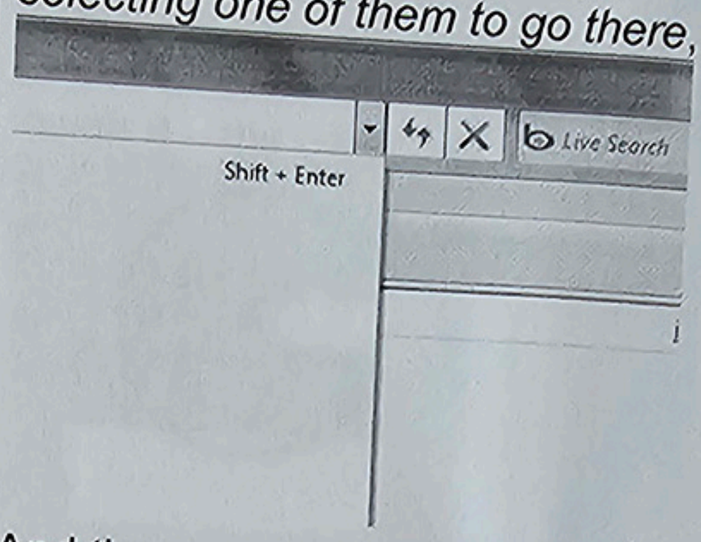
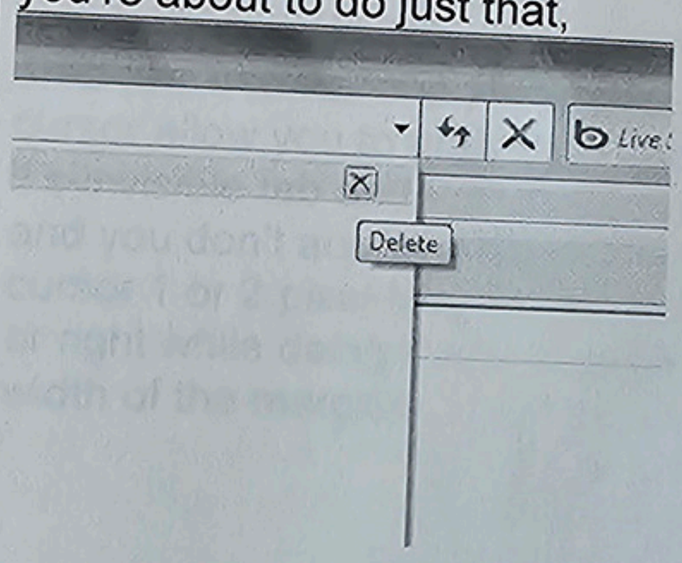
A G

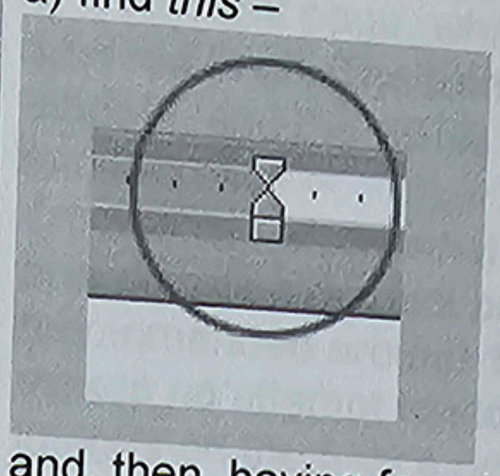
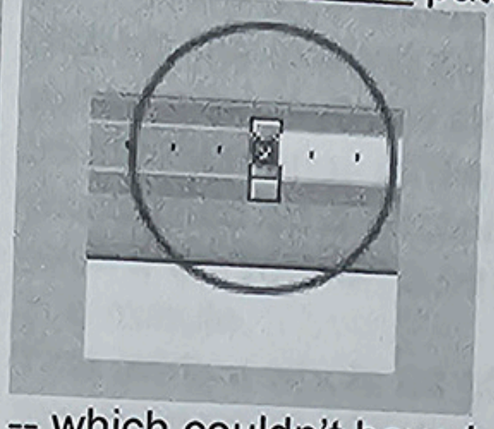
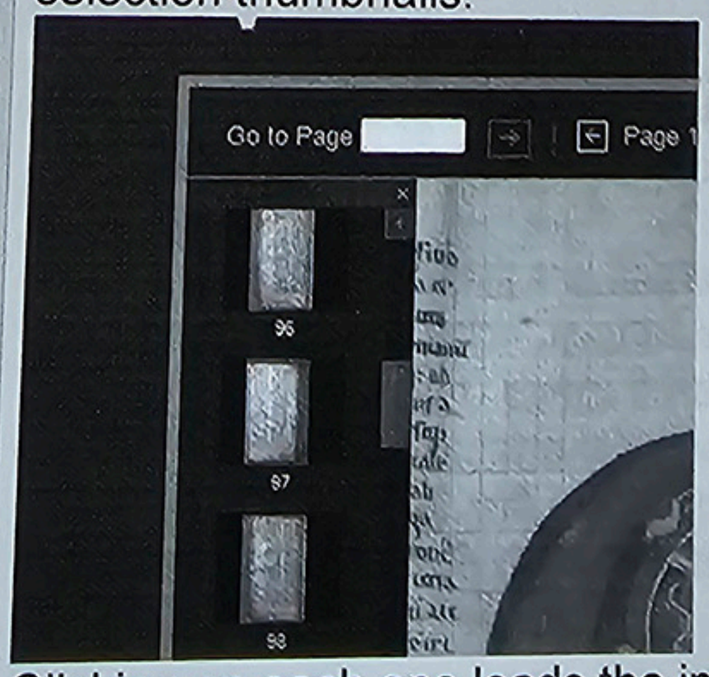
This is a clearinghouse for:
-all notes taken on issues with, and possibilities of, interfaces

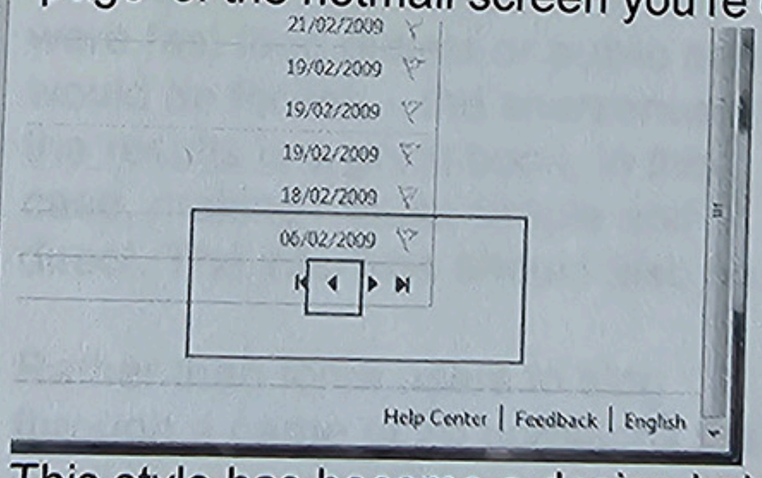
Biggest Pains and Banes in need of Solutions

#	Dilemma	What happens?	How to fix?
01	Coy keys	Like a bashful lover, keys refuse to respond to commands which have been clearly given (i.e. a person's pressing the key) until the user protests several times that they *really did* want to give that command – usually by re-tapping the same key again, when nothing else has changed or elapsed. There is clearly no reason for this, as data entry through the keyboard is practically as old and refined a process as having a CPU in the first place. Still, the computer seems deaf at times, in need of a keyboard hearing aid. Even though computers take direct input, and the mechanism through which the correct input is received is the same one (the keyboard) as the one which misses the correct input – this malfunction of an interface simulates deafness, distraction, or some other form of human-like impairment which, for the computer to have, it would need to be, at the very least, anthropomorphized. Besides, this *never* happens with functions that are deemed *important* in the programmer's eyes – just the less-loved ones, of which there shouldn't be any, in this day and age of efficient data processing.	Value the keyboard as a data entry device that is vital to the program, and treat it accordingly. Have the software *acknowledge* the advances of the user *tout de suite*. Save coyness for the veranda.
02	Clever programmers	Let's admit across the board: Geeky programmers are innately clever. Now let's all be discreet about it. Do you hear, programmers? We *know* you're smart, so please, show it in subtle ways, like getting the job done already. Cleverness should be a sidebar, not something to thrust in the user's face minute by minute. We *know* you have near-Godlike powers to make screen elements hover, detach, re-attach, quiver and glow – but out of context, and in any more than a minimalist way, we don't want to see your grandstanding. The line is crossed when a programmer's cleverness oozes out of the program's functions, and drips indelicately into the lap of the user. Some programmers seem inclined to make the interface *more* cumbersome simply to announce their clever presence – like that handicrafted blue metal	Give users an interface that's clean and smooth, which challenges them to assume that it just cropped up naturally and that the programmer doesn't even exist.

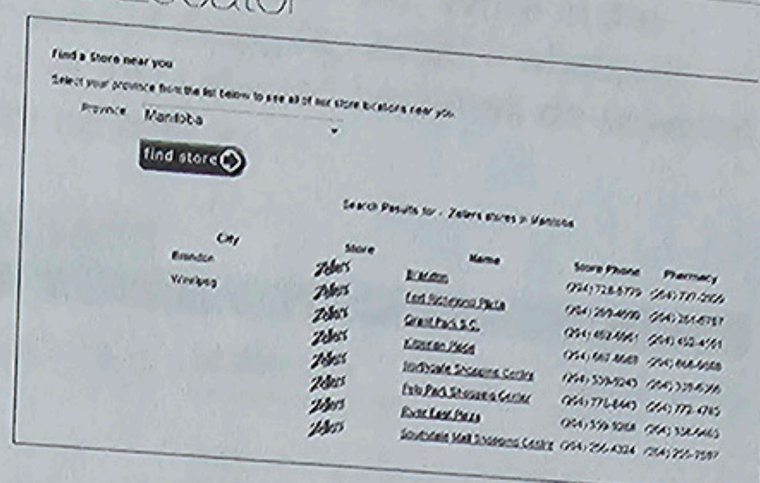
		duck your Aunt Ginny feels a need to hang over the front entrance door.	
03	On-again Off-again fields	What's the one predictable thing a user wants to do when surfing to the baroque homepage of Wikipedia? No, really, think about this for a minute: Hint – it's the *only thing* the user can do on this page, aside from simply regard the page itself. The *one thing* a user will have control over to the exclusion of all else. Anyone who has ever surfed to Wikipedia for any reason will have little trouble telling you: They want to type in search criteria. With no exaggeration, this is the *only* function the user can perform on this page. They can't edit articles – that's further on, when they're actually *at* the article. They can't make suggestions – that would be on some other page as well. No, there is only *one* field that can be populated on the initial Wikipedia page – and that field is the small one on the upper right where search criteria can be input. Too bad the cursor disappears from it almost instantly and the field has to be re-selected by the user, afresh – that is, almost every time the page loads – despite the fact that they are on this page to do *nothing else* but employ that field. Wikipedia, furthermore, can't even fall back on the excuse that the programmers of YouTube might easily give – that the loading of an ad, which happens shortly after the main body of the page loads, somehow provides distraction to the CPU and takes its attention away from the cursor's placement. They can't fall back on this explanation because Wikipedia has no ads. So why the disappearing cursor?	Understand that users want continual access to the *one* task that the page gives them leeway to perform. Keep the cursor in the field until such time as it is used for its intended purpose. *NOTE – (2017) – IT IS TRUE THAT THERE ARE A FEW "CLICKABLE LINKS" ON THE W. HOMEPAGE AS FOR THE IMMEDIATE UTILITY OF THE PAGE TO USERS, THOUGH – ITS VAST MAJORITY IS THE CAPABILITY TO PERFORM A SEARCH. AG
	One-way valves	Sent a pic from cell phone to email address. When arriving, a dumb Microsoft / Telus interface 'displayed' the picture, or half of it, at 'actual size' – meaning that you had to scroll across it to view before downloading. No scroll tabs given, meaning that devious techniques like the 'press on scroll wheel' input have to be used. When doing that, was possible to scroll to *right* on pic, but then when it came time to scroll back to *left* – interface was coy as a pampered lover, and refused to. So: What little use for file was made available, existed in a one-way mode only – with no hint as to a way out.	Something elementary (like scrolling to the right across a page) should be easily reversible, by going back out the way you came in. (I.e. same click, different mouse motion). Allow symmetrical back-and-forth actions where they're not prohibited.

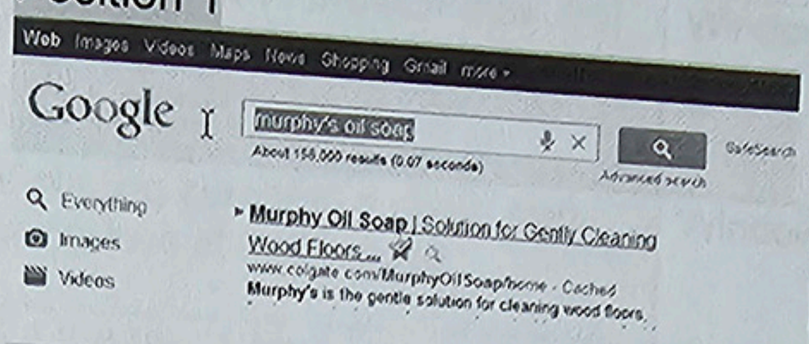
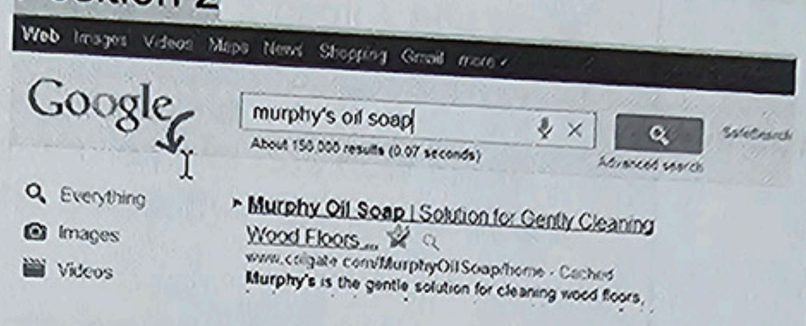
Which interface?	What happens?	Issues / Possibilities?
Microsoft Internet Explorer, ver. 8	When you click on the 'little arrow' to bring a drop-down menu of recently-visited sites, *for the purpose of selecting one of them to go there,*  And then proceed to move cursor to the 'white and clickable area' where you're about to do just that,  Surprise! A previously – *invisible* 'red-x-delete' button appears. Its presence is *a surprise*, since it immediately layers on another – and opposite – function for the exact same white space you planned to use to *surf to* the website. If you click this (by accident or otherwise,) the record of your webpage *is deleted* – therefore making it *very hard* to go to again, especially if you don't have the exact address for it.	The programmers have proven themselves to be *very clever* in making a button appear – Poof! – where nothing was before. In this sense, they are like Jeannie, or some Teenage Witch or other. However – It's a good idea *not* to make buttons appear *which do the opposite of the primary thing the user is trying to do*, especially *without warning* and *in the exact area that the opposite button is supposed to exist.* Seems to be an issue of *boundaries* (between opposite functions), *size* (of teeny buttons that flit in and out) and *functionality* (keep *one function* to *one space*.) Others could possibly be of 'anticipation' – the act of using ramped visual cues to *guide* the eye along the path of an action, helping users *anticipate* what they'll find. (and, no – that does not have to mean 'bouncy ball-y' interfaces where everything seems to be floating pieces of rubber.)
Microsoft Word – 07	You want to shift the margin to the left. Sounds simple. But, to do so with the mouse, you have to	Clearly, the programmers *meant* to give people a way of using the mouse to change margins.

	a) find *this* –  and, then, having found it, b) find a considerably smaller area inside it – the <u>other</u> patch of red –  -- which couldn't have been more than 4 pixels across. -- only *then*, while hovering over that area (the internal one,) will your cursor allow you to grab the so-small-it's-invisible tab and – if you're lucky, and you don't accidentally shift your cursor 1 or 2 pixel-lengths to the left or right while doing that – change the width of the margins.	And it *is* in there – however: It has no graphical representation. (The 'red' was added after.) The only way the user knows they're doing that operation is that the <u>cursor</u> changes to a different icon. And even so, that's only within a radius of about four onscreen pixels – making it a test of gross motor skills every time you do it. The programmers should review which functions they're trying to depict in their graphic user interface – And *depict* them, instead of leaving them as a pinprick – sized hole in the program. The issue here is: Assign a grabbable graphic symbol to each function in a graphic user interface.
Thorough reproduction of Codex Gigas - <u>http://www.wdl.org/en/item/3042/pages.html#volume/1/page/101</u>	A lavish one indeed, it's hard to find fault with it – *however* – in this case the fault arises *from* its own lavishness: On the left side are a series of page selection thumbnails:  Clicking on each one loads the image	This is a case of overprogramming – and, simultaneously, overlooking one of the most simple assets in modern pcs: Their increased memory. Around 1982, when hard drive space was at a premium, it would have made sense to re-load everything from scratch when clicking on any thumbnail. Now, though – it defeats the purpose of ease of use, by denying the user the ability to reference a screen that they *just referenced*, seconds ago.

	of a different page, which can then be zoomed in on. The transition is butter-smooth, and the whole thing follows intuitive lines -- *Trouble is*: It does what some programming manual *probably* recommended around 1982 -- It makes *no* attempt to store any buffer images on the user's drive, for temporary use (like a cached webpage). So – *every* time a thumbnail is clicked, the loading process happens *again*, even if you're simply clicking on the thumbnail of the same image you clicked on a second or two ago – even if it's your 5th time clicking on it in a minute. This adds a whole lot of wait time to the generation of the results, and prohibits users from easily 'flipping back-and-forth' between images. (defeating a big part of the purpose of studying texts.)	This doesn't even come off as an attempt to show off. It just seems sloppy. It's kind of a surprise, considering how much time and effort seems to have gone into the rest of the webpage. The programmer should revise things so as to eliminate this redundancy – and then they'd have something as futuristic and sci-fi influenced as they seem to have succeeded in evoking elsewhere on the site.
Microsoft Hotmail	It's hard to beat Hotmail as a collection of the strange, the gawky, and the overprogrammed. Among their most potent examples of gawky are the 'forward' and 'back' buttons, that are supposed to control which 'page' of the hotmail screen you're on.  This style has become a design habit with Microsoft – but that's no excuse for it. A good analogy for these buttons would be a light switch – that *changes positions* in your room every time you use it. If you turn it on near the door, it disappears from the wall, and – *pop* – re-appears across the room, by the far corner. Of course, in real life, this would be	The solution to this one – and the cure for that familiar, suffocating '*where is it*' -- panic, felt all too often by Microsoft users – would be to *keep the buttons on screen*. Another analogy: A cafeteria hot table, with removable trays. What if the *temperature controls* were welded to the *food trays*, so that every time you re-filled the meatballs, you would have to question *where your temperature dial was* – and, if it had been taken (with the tray) to the kitchen, you'd have to find it and ask permission of whoever was holding it before adjusting the heat. True, some governments have employed policy engineers who came up

	creepy, and, of course, not really useful. It would mean a special trip across the room to 'undo' what you'd just done near the door. That's how these buttons work, though. It's not enough that they've fallen prey to the habit of making them *tiny* – almost as bad as their 'non-buttons' in Microsoft Word – but use one, and *immediately*, it scoots off somewhere beyond your window -- that is, off screen – and, if you want to *continue* in the direction you started, you have to find it again. It's like going through your email in an earthquake. Following this style of design, they might as well take you to Amazon.com after each email, just in case you wanted to shop for gifts.	with cockeyed schemes like these – but usually, they got the job through their uncle or something, and were, in reality, dumb. It would be comforting to know that Microsoft employees needed some prior experience before being hired – and that they could out-think their pet poodles. Not that it would make up for breaches of common sense, though.
Zellers Store Locator	Kudos to one that *works* – and to a company who appreciates that speed and ease = convenience, and that equals a happier customer who cares enough to visit them. It's *so simple* to do it right, you wonder why almost nobody does. The number of Zellers outlets per capita is tiny. To make an interface that searched for stores as if they were fast food outlets or public parks would be foolish – the sparseness of the results is a great boon, in this case, making results simple and direct. The interface should also be. Rather than force users to slog through a game of 20 questions that would be more suitable on a credit card application, it asks *one question* – a fairly minimal one, at that: *What province are you from*? The answer is easily selectable on a drop down menu (just a handful of choices). After that, you get – pure and simple – a *list of stores in your area*, which, through the simple math of the situation, will be easily displayable within a single field of vision on a manageable area of the existing page.	Zellers is in the process of stepping up its game to make something of its massive distribution network, and compete with the big boys – Walmart, mainly – who have run roughshod over territory that Zellers owned prior to the era of Free Trade. Ironically, the people behind this are themselves Americans – but that means that they refuse to be intimidated by the traditional Canadian mindset of relying on artificial government constructs to prop up a given industry, lest it fall to American competition. Here, they soundly trump the American – designed Walmart store locator page, by doing *less* – by figuring out what a satisfying result might be, then not doing a whit *more than they have to* in achieving it.

	 It's taken so much longer to describe it here than to do it. It is a *single click* operation – and while even that might be excessive – it's an indicator that someone there wants your business and is willing to test interfaces as well as design them.	This reassures the customer that economy is the order of the day. It might be too obvious for some, but in cases like this (see Hemingway for more examples) less can truly be more.
Scrubbing Bubbles Extend - A – Clean	Here's another one that definitely works. It's an example of fresh thinking about physical systems. The spray head is detachable from the refills down below, but together they form one complete bottle (design coup #1). In addition, something very unique: All the stressful hand action required to continually squeeze the trigger on this bottle has been replaced with *one* motion. (design coup #2). Hold it down – pressing it once – and a battery - operated mechanism in the dispenser head does the rest.	This was based on a simple analysis of motions in a complete action – the type of which all computer designers should do at some time. The *separate action* of squeezing triggers on bottles of this nature is *at odds with* others – like aiming the bottle, keeping it steady, etc. – What this adds up to is a key part of the use defeating the purpose of some others. This is the exact same thing that happens with the modern mouse's interface, where the *separate act* of 'clicking' naturally *de-stabilizes* every other motion the user has undertaken. The simple solution – mechanize one destabilizing action, allowing it to mesh smoothly with the others – should point the way to many future good interface designs, whether physical or virtual.

Selection made in various fields – featuring the Incredible Disappearing Select Field	Click. Drag. Select? No. While in the process of dragging, *blink!* – whatever you've just selected becomes *de*-selected – for no reason. Position 1  Position 2  	Acknowledge that, when a user is *in the process* of selecting something, that they want to *select* it – and not *not* select it. Let them *continue to* select it, rather than enacting a series of staccato magic tricks onscreen where the select field, oddly, disappears because the select icon moved over not more than half an inch – to repeat – *while* user has continued to select with the mouse.

"tabs to grab" – begun June 2009

Interfaces

Basics -

SIZE
Remember the size of humans, size items accordingly

CLARITY
Avoid ambiguity - tailor cues to acknowledged print iconography
Pare down number of options to what's used - discard those that are never used
Be consistent - make utility of things a given
Use elementary shapes

SPECIFICITY
Don't withhold important data
Offer clear and present handles

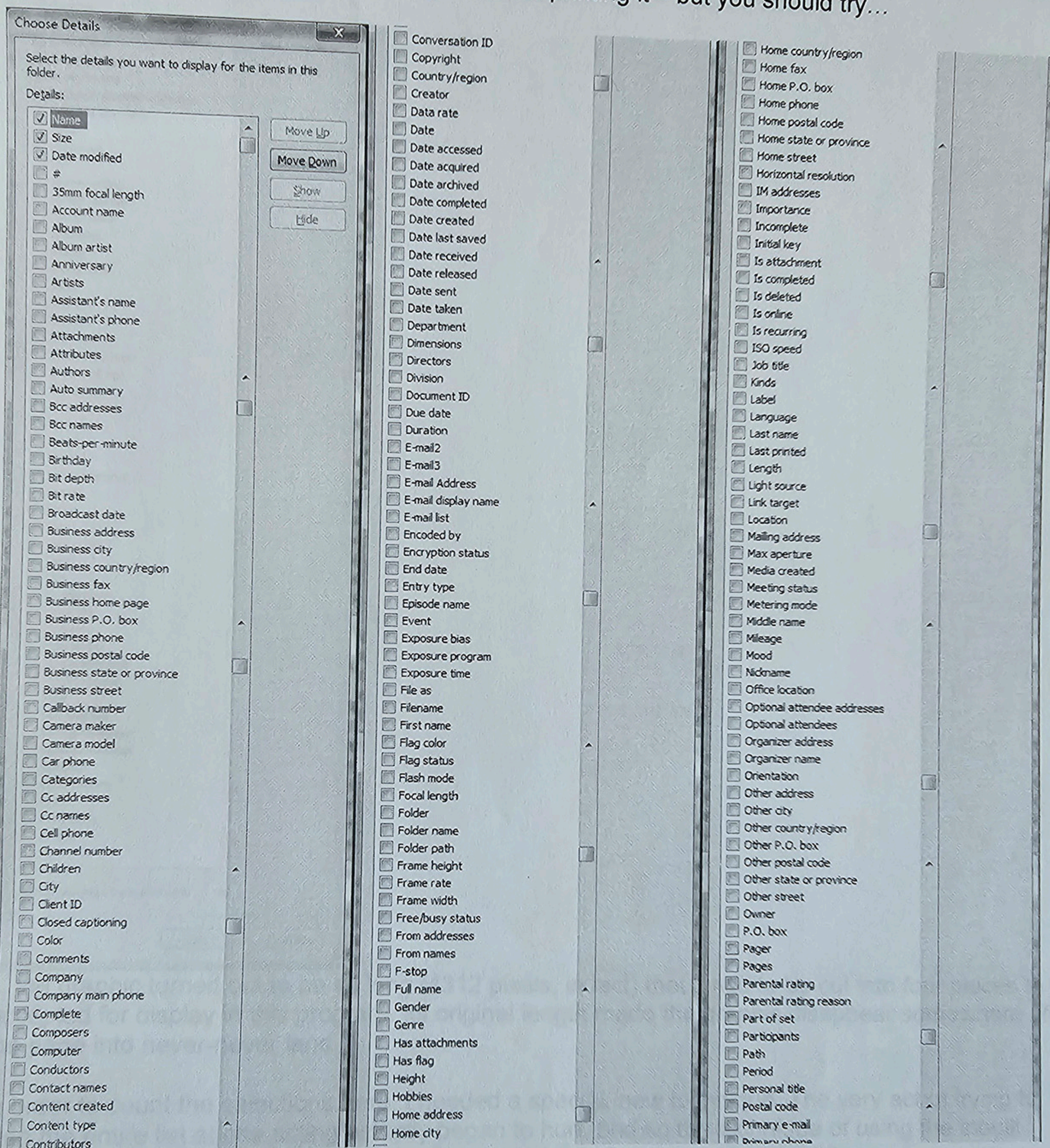
SPEED
Be responsive - microseconds count - treat as 29.97 FPS video

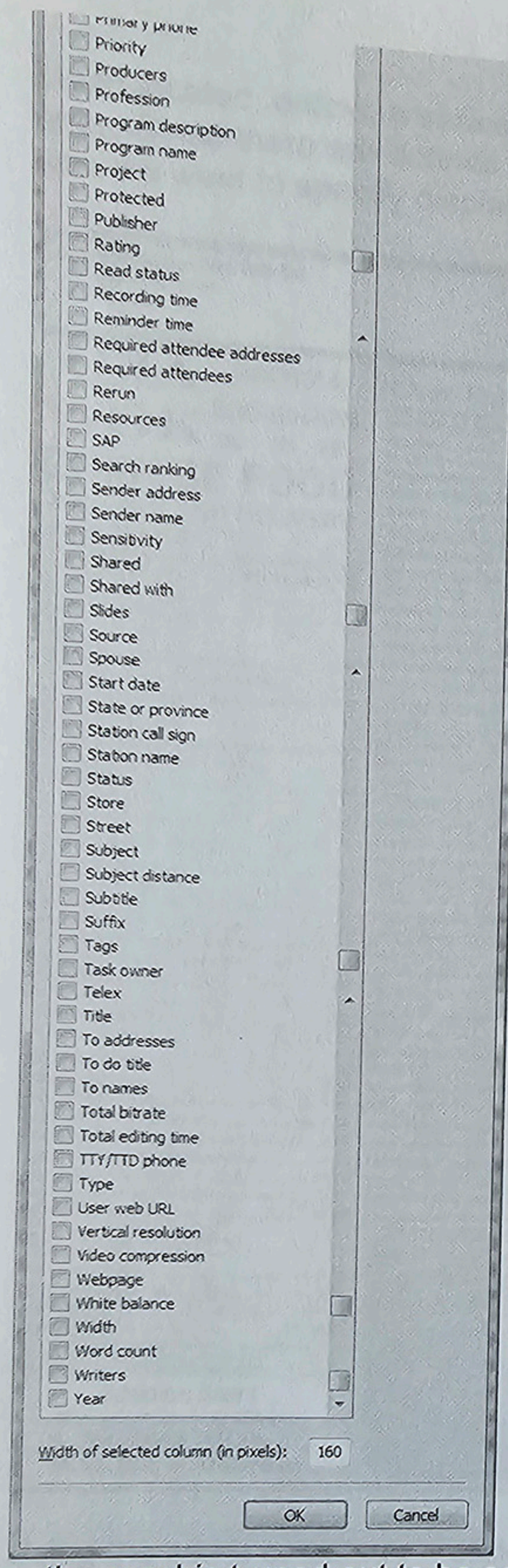
PRESENTATION
Be aware of the aggregate effect of individual small stuff
Go for simple and easy

01	Don't make the tab to grab so insanely tiny that a surgical pass over it with an ordinary mouse results in the cursor only changing to the target indicator icon 50% of the time - the flicker issue	MS Word - Little upper margin tab (4 places on page)
02	Make use of the app's negative space to provide a neutral field in it that's always clickable - so as to give a natural way to de select items	Windows XP window
03	Don't make the function of placing cursor in a given area ambiguous - and make user guess at its purpose	Windows Vista desktop
04	Have a wide berth around a the tab to grab inside which it's grabbable - and show user where it begins and ends	Windows Vista desktop
05	When a chunk of text gets automatically selected, and an arrow key is used, make it non-ambiguous how many spaces on which side of the chunk the next move of the cursor will be selecting	Windows Vista right click
06	The tag issue: Reveal helpful data to user - at least don't sit on potentially important data for no reason, making it invisible, when it's an easily disclosed part of a file's id tags	Windows Media Player: Although the complete running time of a file is, say, 4:04, WMP identifies it as '4 minutes'
07	The speed issue: When the choice is there to respond to user input slow or fast, respond fast	Windows Media Player
08	Make icons resizable - do not have a mess of 5 cent stamps that all resemble each other	Windows desktop
09	The water vs electricity issue - avoid putting widely disparate items shoulder to shoulder - ie do not put the much used copy function right next to the delete function	ACSR and right-click Windows menus which have 'copy' and 'delete' perilously close, in non-re-sizeable tiny type
10	The page number issue - make predictable areas where a given tab to grab can be found - do not have tabs flip and flop here and there all the time	Windows XP system interface
11	The windows issue - where it's possible to concisely summarize a database in one screen, do it	The Apple tree approach to paths
12	Allow user to set consistent defaults. Do not always offer hundreds of randomly displayed defaults just because their potential has been included in programming.	Windows Vista
13	Have the same button do the same thing each time it's pressed	Windows Media Player
14	Minimize annoyances - the clippie issue	Windows XP search
15	Don't have a feature happening regularly that's unpredictable and doesn't appear to have a use	Windows Media Player - the giant note in the righthand corner that randomly resizes itself and variably displays details, sometimes not
16	The tortoise and hare issue: Search for the reasonable	Scroller tab on Windows Vista

	median speed and offer users a middle ground between a slow crawl and a supersonic dart - or, better yet, varying gradations of speed grabbable by a clear and present tab	
17	The rink or rubber issue: Do not produce objects and surfaces that are 'skate-y' - that have the unsettling quality of being a field of squirming protoplasm under the cursor. Instead give surfaces traction like rubber.	Windows Vista left-hand component
18	Don't leave the user scratching their head while they search for ways to do elementary actions like 'scroll right' or 'move up' - telegraph how to do stuff	Windows Vista left-hand component
19	Try and find the 'obvious easiest' method of honing in on something. Don't avoid what's obvious 'because it's so obvious it couldn't be right' - in fact, some things appear obvious because they're right.	The method of center pivot in Flash Earth
20	The Only two Eyes issue: Offer choice, not quandary. Once someone has 'ordered', stop showing them the menu.	Selection methods in almost every Microsoft app
21	Avoid animation that's not called for - simply because an item's adjacent partner closes, doesn't mean the current item should change, shimmer, react, glow, or move	The 'sideways tetris' effect of bottom item bars in Vista
22	Smoothly transition (using POV) by inserting slow-in - slow-out inbetweens into the transition from 'item there' to 'item not there'. Always give smooth cues to the user of what's about to happen or what has.	The new interfaces of Vista
23	Insert the laws of physics into simple interfaces. Using simple geometric shapes as windows, applying physical rules can conjure thousands of inbetweens in real time which would otherwise have to be pondered over in a studio by many planners	Physics Flash games
24	Use principles of motion and visual cues as explored in Norman McLaren's series 'Animated Motion' from the mid-70s	Flash Earth
25	Define the intended outcome - since a given action can consist of many sub tasks. Once defined, string together a connection between initiation (the user signifying what should be done) and achievement of the action string. Once the path is defined, attempt to smooth and make the interface treat this chain as 'one action' with one visual stretching over it.	Harper Lee: '…'d you see him, Scout? 'd you see him just standin' there?...'n' all of a sudden he just relaxed all over, an' it looked like that gun was a part of him…an' he did it so quick, like…I hafta aim for ten minutes 'fore I can hit somethin'… Jem Finch about Marksmanship - chapter 10 TKAM

**About number 12 on list above:
Incredibly, what follows is a graphic of *exactly what the user sees* – or, better put, is confronted with – when petitioning the Windows OS to let them only see certain categories of metadata about a given group of files. Good luck viewing it without printing it – but you should try…





…the graphic turned out to be so long (4312 pixels, in fact) that it had to be cut into four pieces to be accepted for display in this program. Its original length made the graphic disappear somewhere off the page into never-never land.

In order to count the selections here, I needed a special, new technique. The very act of trying to read the entire list at one sitting actually began to hurt, and so the technique of using the inbuilt Windows calculator (coding date circa 1980) as a kind of click-counter was hit on.

Fortunately, being able to click the 'enter' key once for every item on the list I passed over – 'read' is too charitable a term – and then have the numbers (1 + 1 in a sequence) displayed in total for me at the end of the process gave me a usable figure. Any attempt to check the progress of the clicks during the counting would have resulted in losing track of place in the list – so, with the calculator – click method, I was, at least, able to do the tallying without moving attention from the list at any time. Apparently (I'm not doing it again to check) there are 242 selections.

I then located, online, a series of take-out menus – from which I selected one of the denser looking ones. Since there are definite similarities between these menus and the one Windows gives us every time we want to specify display defaults, here's the one found online:

Asia Wok Chinese Food Take-Out
2026 Tenth Line Rd, Orleans
613 837-2233
Cuisine Type : Szechuan
District : Orleans
VISA
ASIA WOK
Cantonese & Szechuan Dishes
亞洲閣
CHINESE FOOD
TAKE-OUT AND DELIVERY
Appetizers
Egg Roll (Close End) (12) 12.00 each 1.20
Egg Roll (Open End) (12) 12.50 each 1.25
Spring Roll 1.85
Vegetable Spring Roll 1.75
Soup
Chicken Noodle Soup, Chicken Rice Soup 3.25
Chinese Vegetable Soup, Mushroom Egg Soup 3.25
Hot & Sour Soup, Won Ton Soup with Bar-B-Q Pork 3.85
Fried Rice
Beef, Bar-B-Q Pork, Bacon Fried Rice, Chicken Fried Rice 5.95
Mushroom Fried Rice, Vegetable Fried Rice 5.95
Plain Fried Rice 5.15
Shrimp Fried Rice 8.95
House Special Fried Rice 8.95
Beef and Pork
Beef with Macaroni 8.50
Bar-B-Q Pork (sliced) 8.50
Beef w/ Green Pepper, Garlic or Sweet & Sour Pork Balls 9.45
Almond Bar-B-Q Pork Ding, Bar-B-Q Pork w/Mixed Vegs 9.45
Beef w/Mixed Veg, Beef with Chinese Green (Bok Choy) 9.45
Bar-B-Q Pork with Broccoli, Beef with Broccoli 9.45
Beef with Mushroom 9.75
Garlic Spare Ribs 10.15
Cantonese Sweet & Sour Pork 11.55
Double Cooked Pork 11.15
Chicken
Chicken with Macaroni 8.50
Almond Chicken Guy Ding, Moo Goo Guy Pan 9.45
Sweet and Sour Chicken Balls, Pineapple Chicken Balls 9.75
Garlic Chicken Balls, Dry Jar Doo Chicken Wings 9.75
Chicken with Broccoli, Garlic Chicken Wings 9.75
Almond Chicken Soo Guy 10.25
Garlic Chicken Soo Guy, Pineapple Chicken Soo Guy 10.25
Mushroom Chicken Soo Guy 10.25
Curried Dishes
with celery, onion and pepper
Curried Mixed Vegetables 8.75
Curried Beef, Chicken, Pork 9.95
Curried Shrimp 11.95
Curried Seafood 12.95
Prices Subject to Change Without Notice, Taxes Not Include
Small Orders 85% of Full Order Price
Hot and Spicy Dishes
Business Hours
Mon-Sat : 11 a.m. to 12 Midnight
Sun : 4:00 p.m. to 11 p.m.
(Min. Order $10.00) - $1.50 Delivery Charge
Chop Suey / Chow Mein
Made with fresh bean sprouts celery, onion and carrot
Chow mein comes with dry noodles 0.30 extra.
Vegetable 7.95
Beef, Bar-B-Q Pork 8.25
Chicken, Mushroom 8.25
Shrimp or House Special 9.45
Scallops 9.95
Vegetables Dishes
Assorted Vegetables in Spicy Sauce 9.10
Almond Vegetable Ding 8.45
Buddha's Delight 9.10
Cantonese Vegetable Chow Mein 9.95
Mixed Vegetables 8.45
Stir Fried Chinese Green (Bok Choy) 8.45
Stir Fried Green Pepper 8.45
Vegetarian Singapore Vermicelli 9.95
Egg Foo Young (Chinese Omelet)
Vegetable 8.95
Beef, Bacon, Bar-B-Q Pork 8.95
Chicken Foo Young, Mushroom Foo Young 8.95
Shrimp 9.75
Chef's Suggestions (No Half Orders)
Beef or Chicken with 3 Kinds of Mushroom 11.75
Beef or Chicken with Cashew Nuts 11.25
Beef or Chicken with snow peas 11.75
Beef or Chicken in Black Bean Sauce 11.25
Shrimp with Black Bean Sauce 11.95
Shrimp and Chicken Delight 11.95
Lemon Chicken 11.95
Happy Family 11.95
Cantonese Sweet & Sour Pork 11.75
Sea Food Delight or Tai Bop Voy 12.95
Seafood (No Half orders)
Almond Shrimp Ding 9.95
Breaded Shrimp (Sweet & Sour or Garlic Sauce) 11.95
Breaded Shrimp w/Mixed Vegetables 11.95
Breaded Butterfly Shrimp with Bacon 11.95
Fried Shrimp w/ Mixed Vegs 11.95
Fresh Shrimp with Broccoli 11.95
Fresh Scallops w/ Vegs 12.95
Fried Shrimp w/ Cashew Nuts 11.95
Shrimp w/ Lobster Sauce 11.95
Sweet and Sour Scallops, Fresh Scallops with Broccoli 12.95
Fried Shrimp with Snow Peas 12.95
Combination Plates
(sorry, no substitution please) One Egg Roll & Chicken Fried Rice with
Combo # 1 9.90
Garlic Spare Ribs, Sweet and Sour Chicken Balls
Combo # 2 9.90
Chicken Chop Suey, Sweet and Sour Chicken Balls
Combo # 3 9.90
Chicken Chop Suey, Garlic Spare Ribs
Combo # 4 9.90
Beef with Mixed Vegetables, Sweet & Sour Chicken Balls
Combo # 5 9.90
Almond Chicken Guy Ding, Sweet and Sour Chicken Balls
Combo # 6 9.90
Almond Chicken Guy Ding, Garlic Spare Ribs
Speciality Noodles (No Half orders)
Cantonese Chow Mein 11.25
Cantonese Vegetables Chow Mein 9.95
Cantonese Chicken Chow Mein 9.95
Cantonese Beef Chow Mein 9.95
Chicken with Flat Rice Noodle w/ Chinese Green (bok choy) 11.25
Beef with Flat Rice Noodle w/ Chinese Green (bok choy) 11.25
Fried Beef with Flat Rice Noodles in Black Bean Sauce 11.25
Fried Beef with Flat Rice Noodles (Dry) 11.25
Sam See Chow Mein 11.25
Shanghai Chow Mein 11.25
Seafood Chow Mein 12.95
Singapore Vermicelli 11.25
Szechuan Noodles with Chicken 11.75
Szechuan Noodles with Beef 11.75
Szechuan Specialties
(Hot and Spicy, No Half Orders)
General Tao's Chicken or Beef 11.75
Hunan Style Spicy Chicken, Hunan Style Spicy Beef 11.25
Kung Pow Chicken, Orange Beef 11.25
Kung Pow Scallops 12.95
Kung Pow Shrimp 11.95
Orange Chicken, Spicy Chicken (dry) 11.75
Crispy Beef, Szechuan Beef 11.75
Szechuan Chicken or Pork 11.75
Spicy Shrimp with Tomato Sauce 11.95
Double Cooked Pork 11.25
Shredded Beef or Chicken with Garlic Sauce 11.25
2026 Tenth Line Rd.
Corner of Innes & Tenth Line, next to CAA
(613) 837-2233
Family Dinners
All Meals come with Almond Cookies & Fortune Cookies
Dinner For Two 'A' 25.25
2 Egg Rolls, Garlic Spare Ribs, Chicken Fried Rice, Chicken Chow Mein
Dinner For Two 'B' 25.75
2 Egg Rolls, Sweet and Sour Chicken Balls, Almond Chicken Guy Ding
Chicken Fried Rice
Dinner For Three 36.50
3 Egg Rolls, Sweet and Sour Chicken Balls, Beef with Mixed Vegetables
Garlic Spare Ribs, Chicken Fried Rice
Dinner For Four 51.90
4 Egg Rolls, Garlic Spare Ribs, Moo Goo Guy Pan, House Special Chow Mein
Sweet and Sour Chicken Balls, 2 Chicken Fried Rice
Dinner For Five 64.90
5 Egg Rolls, Cantonese Chow Mein, Sweet and Sour Chicken Balls, Garlic Spare Ribs
Almond Chicken Guy Ding, Beef with Broccoli, 2 Chicken Fried Rice
Dinner For Six 73.95
6 Egg Rolls, Cantonese Chow Mein, Garlic Spare Ribs, Beef with Broccoli
Breaded Shrimp, Chicken with Cashews, 2 Chicken Fried Rice, General Tsos Chicken

Using the same calculator – click method, I was able to do a rough count – including the specials – and got a total of 109 items.

Nobody, I think, is really expected to take note of *every* featured item on such a menu. At best, I believe, a customer makes use of the helpful categories provided, ruling out certain dishes that they won't be ordering, and examining more closely the categories that look more likely. No such category system is offered in the data dump given by Windows.

So, when comparing between a take out menu with 109 selections, and the Windows menu with 242 – the natural question arises:

Why only 242?

This menu, as given by Windows, is a poor approximation of *all the choices out there for everything in the world.* While it might take days to scan and understand the whole list, I was able to discern

that the menu is *unlimited* in its scope – in that it encompasses data that have import not only for the casual user, but for *anybody anywhere* who might want to look up *anything* – from the initial key to the gender – whether or not any kind of metadata on any of the selectable items is encoded in a file.

Considering that this list must appeal to every global citizen, regardless of purpose or creed – there are some answers Microsoft has to provide. What happened to the selections for…

Blood type
Circumference of bolt
Employee ID
Flavor
Gear
Kennel number
Latitude
Mood of office
Payload
Temperature
Translucency
Trim
Type of meat

and roughly 450 other possible ones?

Then again, space may have been a consideration. Probably not, but what if Bill Gates – or some project director – had set 242 items as the comfortable limit for this selection list? To include some of the more appropriate criteria, such as Gear and Payload, some selections might have to be called into question for possible elimination. Using more intensive scrutiny of the list, the following might require some clarification:

Company Main Phone
E-mail3
Hobbies
Home P.O. box
Parental rating reason
Spouse
Telex

…all of which were *included* in the list of things to be displayed – and some of which will, owing to the willy-nilly no-defaults-allowed nature of this interface, be displayed *automatically*, whether the user needs them or not.

The next natural question coming up: Where, in a file or the information attached to a file, would the Parental rating reason usually be? What are the standard set of terms governing the assessment of it? In other words, how would it look, should anyone actually be able to access a list of it, for whatever reason?

What would a list of the Telex properties of a series of files look like? What is the telex status of, say, that picture of my grandmother by the beach house from our vacation?

How would most people classify the Hobbies of, say, a fourth quarter shareholders' report?

What's the Company main phone of my PowerPoint project?

Is there any guide to any of the above?

Nope.

Should there be?

Well… probably not. A guide would involve a short description of the properties and nature of each categorical selection possibility. With 8 explained per page, this would be something like a 30 page document – which wouldn't be needed, anyway, if Microsoft would let us see anything less than the full 242-item list every single time a user wants to specify display defaults.

Out of curiosity, while on a trip to the computer store with my PC, I asked the repair guy at the main counter – a friendly enough individual – to explain how to do this. Since this feature of the Windows OS had been with it for at least three versions, and possibly more, I assumed that the issue of how to set display defaults for the average folder might have come up – if not provided fuel for some kind of raging controversy resulting in Microsoft fielding threats to firebomb their offices.

What the repair person told me was that he was as confused as I was. He had never, he said, had any reason to think about it. In fact, he'd never noticed this 242-item menu, and, after some fiddling, pronounced it unchangeable – there would be no defaults set by this repair team.

So far, no amount of consultation with internet chat sites has yielded any solutions. That's not surprising, though – because the terms to describe this problem have been my own. Windows offers no list of possible ways to describe parts of its interfaces for users, let alone a guide to modifying them. The terms themselves to even begin to describe aspects of the interface which might need changing are, to this point, secret.

1 '**Hot to Cold**' –

To track changes for the user, what about using the fanciful principle of *friction* in the interface? That is – make the things that change '**hot**' – ('glowing red' or some similar color) – as soon as they settle, they become 'orange' – 'yellow' – maybe in varying gradations –

The net effect of this is to immediately show user what's been changed recently, in color code. Even more: A 'visual diary' – consisting of a shot of the activity of each day in changes – (flippable, so that user can reference quickly between days) – changes in 'heat and friction' made easily visible –

The way this would happen is that items could have a predictable 'half-life' – or just 'cool-off period' – say, a day for yellow, 2 days for beige… or whatever: Basically, though, whatever was 'hot' would show up as one color at the end of one day, then gradually 'cool' unless moved again, so that in the 'daily shot' the user would see just what items are hot – and, flipping between days, would see which items are more *consistently* hot and changing… Think of a meteor tail, but in slow motion –

2 **Behind Microsoft Development Doors – the Genesis of Bloatware**

1st Microsoft developer: *Hey, why do you always have those little random menus popping up whenever the cursor goes over anything? Don't they get in the way?*

2nd Microsoft developer: *Yeah, but we put some effort into programming those things. If they didn't pop up, nobody would ever use them.*

Moral: Just because you programmed something, doesn't mean it's important.

3 No-Click Selection: Motion, Graphics and Text Entry

It's come up that where selecting an icon onscreen is concerned, there's 'more than one way to skin a cat'. This is especially true when a string or chain of choices is presented. In fact – the 'ordinary' way ('clicking' on something) is really just one of many, and *by no means* the most efficient.

The fact is: If the range of selectable items on a menu is relatively limited – that is, if the full array of modular selections can be easily displayed within the field of vision available on one screen – then there's no reason at all to deny a user the most accessible method of mouse-based selection (adaptable to almost any other graphic means of cursor control): The motion-based navigation method.

The *purpose* of a mouse click is roughly the purpose of putting on flashers in traffic: You are *signaling your intent* to another party – in this case, the inner workings of your computer's user interface. The *action* of *very slightly* pressing down on a hair-trigger mouse button has nothing to do with what you *signal* – often what you're *showing intent* to 'do' is just to *progress to a different part of the interface* – again, like traffic signaling.

If the user's *mouse movements* don't affect the 'link' being clicked on (really, a *portal* to the next step in the program,) there's no reason they can't be used to *control* the user's relation to the link. This may not be the most obvious thing to a ground-level programmer – it's much easier to think in terms of 'when x happens (the input of binary data from the click) then y happens (the user 'enters' the area of the program related to the link.)'

A 'click' (from the programmer's point of view) exists in a perfect state of binary clarity: Either it *has* happened or it *hasn't*, therefore clearing up any hazy areas that might make it more awkward to enter in as a program.

However: For the user, something as strict as the clicking of a small device represents a slightly cumbersome move – at least something that requires another level of motor skill *beyond* that of navigating the mouse on a 2-dimensional surface.

It adds a *vertical* aspect to the formerly horizontal plane of movement. Taken individually, mouse clicks may not be that big a deal – *but* when the user is confronted with having to initiate this less than intuitive action *every time* they want to 'go somewhere' in the two-dimensional world of graphic links, it all adds up and starts *unconsciously* feeling chore-like – even to the point of annoyance.

The real question is: What is the easiest way? It's time for programmers to start thinking in terms of *smooth motion*, instead of simple binary relationships. The closest analogy is that of play structures found in a kid's park.

Realistically, the computer screen can be seen as a bit of a game. All graphic user interfaces have that quality to them. In a series of children's play structures, the click is a lot like stepping stones (a series of short punctuations) and a motion – based interaction is like a kid's slide.

Let's say you're an adult, trying out this equipment for old time's sake. It would hardly be tolerable to stick to one or the other piece of equipment every time you went to a certain park. However – between the two, which would you choose? Let's say you could go to a park with *only* stepping stones or *only* the slide?

The basis for comparison would be simple: Do you want an exacting, choppy, contemplative activity where you're always thinking about your next move, or a smooth, contiguous ride where you

experience the thrill of fast motion and get to your goal in far less time? Of course, you might *prefer* one or the other, maybe out of habit or nostalgia. But – where *time* is a factor – you'll always choose the slide, since we hate to be held up more than we have to. At least the option should be there.

There are some intriguing forays into motion-selection on the web already but what'd really be an excellent display would be to make an experimental, *full text entry* interface – which allows users to enter either chunks of text or just one alphabetical letter after another – where speed, motion, and *the relation of the cursor to the graphically depicted dimensions of a given icon* are the only methods of data entry. Say, a 'Wikipedia' – style full-text index, with the navigation being done from letter-to-letter click free. It would give the programmer cause to show off some of the true higher functions.

It should be noted that the *purpose* of data entry in a Google – style *search* is *very* different from any kind of continuous textual writing. Unless the user is a professional typist, the 'search' procedure is a lot more casual than anything else – and, furthermore, it's done *in conjunction* with mouse movements, more often than not – the user will select search results using the mouse. A mouse-based approach to entering simple (or even complex) search criteria would allow the nature of this new form of text to come into its own, or, at the very least, unite with its helper tool, the mouse.

We've become accustomed to keyboards as the three – dimensional items they are. We've even tried, without much success, to re – design the fundamental keyboard into something flat. While this might be the only solution in things like iPads – it's far from perfect.

We extend the failure to develop keys as 2D objects, though, into a false generalization: We fool ourselves into believing that no 2D on – screen data entry could be efficient.

What we should try, though, is the fluid – motion approach that some far – seeing programmers built into the Start menu in Windows. The 2D tools offered by the motion of a small point (i.e. a cursor) across the dimensions of onscreen icons are more than sufficient to communicate almost any sequence of letters, *especially* when dealing with *search terms*, whose length will be short by nature.

Here's a dramatization, using text levels to simulate moves of the cursor –

ABCDEF**G**HIJKLMNOPQRSTUVWXYZ	g**mail…**
----------→	
ABCDEFGHIJKLMNOPQ**R**STUVWXYZ	gr**eyhound…**
←----------	
ABCD**E**FGHIJKLMNOPQRSTUVWXYZ	gre**yhound…**
|	
ABCD**E**FGHIJKLMNOPQRSTUVWXYZ	gree**n zone…**
----------→	
ABCDEFGHIJKLM**N**OPQRSTUVWXYZ	green **zone…**
←--------	
ABCDEF**G**HIJKLMNOPQRSTUVWXYZ	greenga**te…**
----------→	
ABCDEFGHIJKLMNOPQ**R**STUVWXYZ	greengr**ass…**
←--	
ABCDEFGHIJKLMN**O**PQRSTUVWXYZ	
----------→	(**greengrocer**) – correct term – click on result

This method would be as fluid as moving the cursor on a page, and would 'type' the entire word *without* the need for anything but the picture plane – total clicks: 1.
Right now the assumption is that the motion of cursor *intersecting* the 'letter strip' would register selection of a given letter. There are a few ways this could happen, including a 'fisheye lens' look for selectables, or a 'H –

lock' function that used the left mouse button (more clicking). The selecter (green) might *follow* the horizontal motion of cursor while remaining fixed to the strip, or even 'slow down' the closer (in pixels) the cursor got to the letter strip. Basically, though, it would be an animated way of no – click specification.

P.S. – one possible way of enhancing the user's impression of which letter is under scrutiny, and ready to be selected, would be the Apple – style fish eye lens approach – where the fish eye effect follows whatever vertical axis the user's cursor is on at the time. (i.e. ABCDEFGHIJKLMNOPQRSTUVWXYZ)... that way all the user would have to do would be to 'hover' somewhere around the list to select, then to 'slash through' the line of type where a given letter was highlighted, to go through the instant action of typing it – no click. The main reason for the fish eye effect would be to allow maximum leeway on the user's part to effectively 'type' a letter – instead of the ones surrounding it. Upon the slash action, some kind of visual follow-through could happen i.e. the letter glowing for less than a second, or something.

(Aug. 1, 2011) – I don't see why the simple *de-overlapping* of the user's cursor from the body of whatever letter had just been selected wouldn't be enough to 'register' it as having been typed as far as the computer was concerned. That'd mean that the user would simply have to touch the letter with the cursor, and then just *slide the cursor off* the letter to select it; whatever way the cursor's slide might happen wouldn't matter as long as it didn't touch other letters beforehand. It'd need beta testing, but this is a possible way it might be made to work.

4 – Why?

Ignorance has a huge, dumb buddy: The question 'why'.
Whenever anyone's too lazy to think around corners, or to respond to a request – out comes 'why', ready to play the heavy and back up the ignorance before a reason can be given.

Trouble is, 'why' usually hangs around with people who should know better – who really shouldn't be afraid of the outside world. It hangs around people who might easily use their talents to come up with something helpful or revolutionary, if they avoided that toxic friendship and instead became friends with the cousin of 'why' -- the more refined 'why not'.

Watch out when you hear someone calling its name: "Hey, I know it might take an extra half day, but you really should program a user-friendly tab in there, for this popular function." "Why…" – It's a very bad habit.

5 – Size Matters

There's an overwhelming trend that comes out to a user when analyzing products from Microsoft: *The company doesn't usually seem to test their own products*. Or, if they do, nobody does anything with the test results.

Either way, among the numerous issues this causes, the biggest – ironically – seems to be the epidemic of *tininess* out there.

As everyone who saw Tron knows, in the computer world, small can seem huge, and the tiniest millisecond can feel like all day – (if you're Tron in a throwing – disc battle with Sark, that is.)

Maybe Microsoft's programmers live a little too deeply in the world of Tron. Or maybe it's something else – however, they could benefit from more attention to the word 'interface', when they design one.

The 'inter' part of it stands for 'interaction' – their programs do interact with people – but it's the 'face' aspect that needs more emphasis.

When you use ‘face’ as a poetic term to describe the actual, physical attribute where things interact, you come close to understanding the uses of the program. Here, ‘face’ seems to be a shortened version of ‘surface’ – (in a similar way to how ‘havior’ is used in Romeo and Juliet – ‘…thou mayst think my haviour light; / But trust me, gentleman, I'll prove more true / Than those that have more cunning to be strange. ‘ – to describe ‘behavior’) – and, while ‘surface’ might describe things, there’s another reason for shortening it: User-friendly programs are aimed at people – with faces.

Here you can sense the responsibility of their designers: To give a user something that *looks him or her in the face*, and can relate to them on a human scale – the way tools have, throughout the history of design from the very beginning.

It’s not enough that a tool has a function to perform *related to other tools* – to qualify as an interface, it has to relate to the scale of the humans who use it. There have to be open and visible surfaces, and things to grab that are grabbable and whose purpose can be deduced without too much hassle.

This isn’t to say that everything has to be *pretty*, or even dumbed – down – as many programmers will argue is the pitfall, when making things convenient – people can handle a little complexity, and they like shortcuts, once they get the feel of a program – *but shortcuts don’t have to be physically short*, and in the human world – unlike Tron’s – *small is not always big*, and in fact, is usually small.

Tron represents a secret portal to another dimension, where people can loose the bonds of reality and exist in a computer – generated world of tomorrow. This can be problematic for regular users, though, because they, as humans, are trapped here *in* reality – and they really do appreciate a program which treats them as full and adult – sized, people who haven’t quite adjusted to life inside Tron’s binary coliseum. People still need people things.

6 - Pickets

If I could picket the offices of those responsible – (who?) I’d be pushing for certain demands. These would relate to certain features that have been AWOL on all interfaces that I know of --- *especially* those that would benefit from it most, like word processors and web browsers – *like, forever* – and something must be done! The human race has suffered long enough!

Cliché buttons

The original meaning of ‘cliché’ is the same as ‘stereotype’ – even though they’ve taken on two separate meanings now – these were chunks of moveable type, each containing a phrase. The (very sensible) idea was that certain phrases *were* going to be employed time and again within the printing trade – and so, in order to save time and effort, the typesetter would have that exact phrase all set up and ready to go, instead of having to re-assemble it.

Well, a long, long time down the road, we have the perfect set of tools to do that for ourselves – and we still haven’t done it.

Of course, there are some built – up prejudices around both words. You really don’t have to call them that, though. You could call them ‘pling-buttons’ or ‘wungles’ or ‘flimps’ or whatever else. It’s a sad case of backward thinking when people deny themselves good ideas simply because an archaically – tied term for them has current negative connotations. It’s this stickiness of language that we still seem to be struggling with.

The thing is – what I’d really like to see is a button: ‘Set Up Buttons”. click it.

So then it asks you for the phrase (pasteable or typeable) which you'd like to assign to a button. You enter it. Let's just assume it's happened 4 times. You'll see this:

went to Loblaw's
The President of Silesia
Jesse Kerrington
the textbook on particle physics

…or something a little more formal – looking, floating above things, off to the side a bit.(or wherever you drag the panel with your cursor).

and *yes*…if you're thinking "hey, that'd mean that a click of any of those buttons would deposit its phrase right into my current window, at the location of the cursor, in whatever font or format I'm using now, thereby saving time" – well, you'd be right! That's exactly what they'd do.

And anything at all, that can help smooth out the bloated, icon-heavy, flighty, tenuous, skate-y nature of a word processor, where copying and pasting is often made as much of a chore as tweezing cake crumbs out of a velvet blanket – would probably be welcome.

Customizeable menus that listen to preferences

Customizeable: Something with the built-in ability to be customized. (Once again, I feel the weight of the priests of the permissive society, calling out 'customized? Don't you mean colonized? Nobody, not man, woman, animal, plant, furniture, spoon, or dust, should be customized. Not only does it go against nature, but it's too much like my upbringing! Mumsy and Dadsy would never let me be me. Well I'm going to! You can't customize me, you stupid fascist!')

Still and all, things should be customizeable. Doesn't matter who told who in Microsoft that it wasn't good. They're wrong. A simple example of the alternate case:

With *no exaggeration*, here's what happens *every time* I right-click on a highlighted chunk of *anything* in Microsoft Word – a menu, about the dimensions of a big supermarket flyer, pops up – which contains:

The font
The font size
"make capital"?
"make lowercase"?
A formatting gallery
a broom
"make bold"?
"make italic"?
"center text alignment"?
highlight color
text color
decrease indent
increase indent
bullets
…wait, that's just the small bar on top. Also…

Cut
Copy
paste
font… (note: already offered up top)
paragraph…
bullets (note: already offered up top)

Numbering
Hyperlink
Look up…
Synonyms
Translate
Styles

…out of which, “Bullets”, “Numbering”, “Synonyms”, “Translate”, and “Styles”…all come equipped with arrows for sub-menus, no doubt leading to a different range of choices.

This is human liberty pushed to its most extreme. In fact, pushed past. If this section of the interface were a child, it’d be Richie Rich. If it were a postal worker with grievances, it would long ago have opened fire on the sorting room. We are witnessing the dark side of programming.

At the most optimistic, *maybe* some good – natured person inside Microsoft had prepared the menu as the *upper* menu, unaware that it wasn’t extensive enough, and only at the last minute did it get re-assigned to the right – click selection on anything highlighted.

However, it’s probably a little more planned than that.

Whoever planned it, though, must have actually decided to give us the equivalent of a small top – panel full of choices, *including* two that are *repeated*, just for good measure – every time we right click on a highlighted item, for whatever reason.

This is like asking your mom for a snack after school, and, each time and without fail, being presented with a plate containing:

four sandwiches
half a roast chicken
spinach quiche
fingerbowls containing cucumber, celery, carrot, corn, and cobb salads, plus sliced radishes, wasabi peas, and 4 types of custard
eggplant lasagna with fresh – grated parmesan, and sautéed mushrooms on the side
3 Swanson dinners, each with a meat entrée, steamed vegetables, mashed potatoes and dessert
beef stew
curried lamb
braised octopus
more curried lamb
3 slices of French tourtiere, each with slightly different filling
2 pieces of layer cake

…and this, after doing it for a few months, and knowing that you *only ever* took one sandwich.

you’d think your mom had something against you – if you could get your mind around any of it. Desiring a simple sandwich and being given a glorious buffet is *not* an amenity of an affluent society, whose every need can be filled. Rather, it’s stupid. It makes people feel dumb for wanting a sandwich and has a sarcastic feel to it, not a helpful one.

When choices are so wide-ranging that the person choosing is affronted or baffled, it has the same bad effect as not enough.

There’s such a thing as *sarcastic* good will – and, too often, in the past couple of decades, it’s shown itself. This is a fairly new kind of defense against the slight impression that you shouldn’t be giving so much – you reflexively give *too* much. Call it the plenty sickness, or something.

Anyway, you would think that the answer to this, in the world of programming, would be to allow people to tell the program *not* to offer 150 options when 3 will do. This *is*, after all, only a small *sub*-menu. All of its contents *also* appear on the menu at the top.

Well, good luck with that one. And good luck finding many actual competitors with Microsoft's word processing program – there *was* WordPerfect, but what happened to them?

Anyway, we may have to resign ourselves to accepting the whims of whatever levels of desk sitters Microsoft has to offer, for most of the rest of our lives – but it doesn't have to be that way with other programs, from other companies.

The simple solution, once again, would be to let the user have the *choice* of telling the computer: "Please only display the 3 options I want on this menu, which I will pick out of your list. After such time as I pick them, please don't make me *re*-pick them every time I open the program."

Give users a choice about their range of choices.

That would be *real* choice.

A Tool Belt

Let the user make a 'tool belt' – a collection of a bunch of most – performed tasks in a cluster, on an easily – clickable menu that could be moved – something like that menu of cliché tools.

I know why nobody has, though – Microsoft hires programmers to *program things*, not to 'remove' bits of programming from the mix. Somebody at the company seems to have instilled real pride in them for what they do. So much pride, in fact – that if anyone *suggested* that people weren't *using* one or more of the range of tools they display right at the top of every window – they'd possibly respond "well, shows how much they know! They've got to use them".

Since there's no feedback mechanism built into a program like Microsoft Word, (unlike Google), where certain users are polled from a web – based source, *not* to monitor what they use the program for, but instead which features are getting used – Microsoft programmers exist in a kind of bubble, outside of which is a rundown of features like so:

unused unused unused unused unused unused unused unused unused used unused unused unused unused used
unused unused used unused unused used unused unused unused unused unused unused unused unused unused
unused used unused used unused unused unused unused unused unused unused used unused unused unused

If anyone was serious about paring down bloated programming, this would be extremely helpful. I'm not sure those at Microsoft really care, though. I guess, lately, this phenomenon has come under study in the form of 'hoarding'. It's still very little understood, though.

Anyway, a 'tool belt' would let the user pick out those things that the graph would call 'used' – and put them in one place.

As a sidebar, if Microsoft were serious about finding out just *what* functions of their programs are actually used or useful, and thereby deserve a place on a program's ribbons – to say nothing of what *chains* of commands users are fond of employing in rapid succession, and with what frequency – they might want to put this simple fix into use, even for a few days:

Advertise a beta testing website where users can go to access cloud-stored versions of their most popular programs – the word processing ones being the most obvious, and the easiest to get data

going on – and make it so that users can log on for certain periods of time (or as much as they want) to use these online versions of software just as they would use the computer-installed ones.

Whatever was done with these cloud versions of the programs would be available to save as a standard file to their computer, as it would normally be if the program were used in a home version. Needless to say, if these versions existed merely on a server somewhere, it would be at the company's discretion to decide when the term of service would be extended up to.

Have simple feedback patrol programs monitor their keystrokes and / or mouse clicks. Once this raw data set is amassed, put it through various kinds of analysis and tabulation. Measure which functions are used, and how. Some would be exponentially more popular than others. Without worry over hurting the feelings of strings of code (since they are inanimate objects or data patterns) programmers could then retain what's popular and hide others. Problem solved.

Defaults

A default is:
"a selection automatically used by a computer program in the absence of a choice made by the user"

(source: Merriam – Webster online)

Well, that definition is only partly correct. In the old days (or the current Microsoft era), 'defaults' were things automatically done by the computer program.

However, this reduces programs to the level of reading a book – in a book, the default is whatever's on each page – and there's no way to change it, except by scribbling notes at the side. (and that's not even changing it).

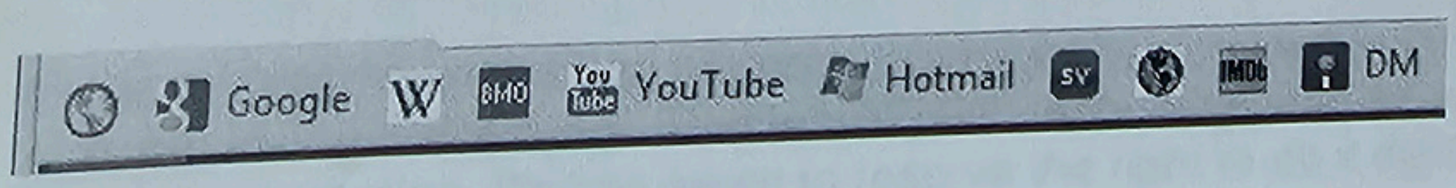
Now, however, with the advent of 'user input', programs can respond in a variety of ways to the user. The dictionary definition can be expanded, in the sense that *some* of these 'automatically used' selections *could* be the result of pre – existing choices made by the user.

This would make the assumption that the user knows what they want – not always a true one, to be sure – but where it *is* true (and evidence suggests that's becoming an increasing number of cases) it would save the user the heart – rending task of having to re-instruct their program to do the same mundane thing, or to use the same settings you carefully set the last time, every single time the program is re – activated, as if the program had alzheimer's, or something – and could never remember anything.

Allowing the user this choice would have the general effect of making everyone happier.

Will it happen? Well, if Microsoft's stranglehold on the industry is ever loosed… it might. Someday, in the deep, deep, wondrous beyond.

***(Aug. 1, 2011) – Well, well! **Cliché buttons** have now appeared on the scene, though not in exactly the way picketed for. What Google seems to have made available, through its new Chrome browser, are clickable virtual buttons on a kind of 'tool belt' – a highly accessible 'strip' of icons across the very top of the web browser's window.



The data set used for their appearance is derived from crowdsourced – that is, web-designer sourced – miniature icons assigned as visual symbols, by the web designer, to a given web page. These symbols are the same that have appeared typically on one side of the web address in the URL bar, as symbols of the current page.

Each comes with a web-designer sourced 'description' – a verbal tag, whether extremely short, extremely long, or something in between – which is editable by the user, resulting in a bare icon or one with a variable descriptive tag next to it.

In fact, these *do* function exactly as cliché buttons, in that the act of clicking once on them triggers the depositing of a string of text in a part of the program – however, it's limited to depositing URLs in the appropriate field to induce the browser to surf to their corresponding page.

Google has only to create a detachable tool-bar of this type, which deposits pre-programmed text strings into fields on a page other than the URL field – and then cliché buttons would be here!

7 – To Be Human

The people at Google Earth seem to want to cover all the bases, but one hasn't been touched. It relates to something about the human brain that we really should understand, if we don't want to constantly sell ourselves short.

The programmers at Google make the mistake of assuming that humans can only progress from *all specifics* to *more general* – that we need to know details before we can comprehend anything about the big picture. This is wrong, and confuses the brain with a PC, but the fallacy comes up almost any time matters of size and scale are introduced.

The simple fact is that humans have, built in to their brains, a faculty that we haven't been able to duplicate in our computer slaves: That of being able to extrapolate detail from generalization.

We've already gone the reverse route of trying to *make* computers do that – such as in the case of the TV manufacturer (was it American?) whose marketers trumpeted, in the mid – 80's, that amazing new digital technology would be able to make a *hi-res* image out of a *low-res* one – simply by applying ('infallible') *extrapolation algorithms* to the digitally processed picture.

This was the poetic dream of technologists ever since, or possibly even before, Ridley Scott's brilliantly – presented Blade Runner posited similar technology in 1982.
In the film, Rick Deckard, a futuristic cop, casually activates a high – tech picture on a (presumably digital) viewing device, and looks at the fine details. Even though the naked eye sees only a room, the digital interface, at Deckard's command, isolates, zooms – in on, and 'extrapolates' fine details – to the point of 'seeing around corners' to hidden areas of the photo – and, just like magic, generating the key piece of microscopic visually based evidence that Deckard is after.

The implication (by Scott and the special effects team) is that this will count as *evidence* for Deckard – or at least be enough to base an investigation on – because this cyber – extrapolation is based on *irrefutable* (and somehow obvious) mathematical processes which appropriate nature's methods, and make the same picture 'nature' would have delivered to the camera's eye, had the photographer been keen enough to seek it out in the first place.

In the wildest dreams of the futurists, this might be achievable through some kind of room – encompassing holography. That would push technology into an analog realm, past the bounds of binary computation. People seem to reserve the right to do it digitally, though.

Since digital imaging in films was only getting off the ground in 1982, when the movie came out, Scott and the effects people had to resort to optical printing and other conventional photographic effects to create this illusion. It was pulled off fancifully and almost seamlessly, and so duped most viewers into thinking the technology might actually be around the corner (which was their intention).

The actual technique is a kind of hybrid of matte effects, a slide show and animation, using photographs at varying levels of zoom and focus as its raw material. Ultimately, what the filmmakers did to simulate detail extrapolation was *1st shoot the fine details, then take other shots from further back*. The illusion stemmed from knitting these still photos together into one effect.

Not only would 'true' digital image extrapolation, relied apon as a reality synthesis, take a lot of the fun out of life, it would also be deadly in some ways.

In the seamless world of Deckard's investigation, humans have easily surmounted all problems of artificial intelligence. The trick, for them, is to sort out which is the artificial and which is the human intelligence, with much debate about what the actual difference is.

In our imperfect world, though, even the *slightest* glitch in reality extrapolation – if humans put all faith in it, like they do in, say, the technology of flight – would mean that a computer – made *lie* would be accepted as truth. Glitches do occur, even if they can be minimized. What we would lose in trusting any such system is our foothold on truth, which, while overrated, is still highly important.

Fast – thinking ad executives, though, will drool at the suggestion that someday, we'll be able to take that picture of our family, fiddle with the zoom settings on it and apply a little algorithmic pixie dust, and in seconds – voila – find Aunt Ernestine's dentures *under the sofa* where she thought they might be but couldn't convince anyone to stick their hand in to check.

It's foolish, though, to send robots out to do what humans have built into their brains already. Maybe the movers and shakers in the programming world became shell - shocked after finding Ridley Scott's vision to be mere fantasy, and, dejected, thought that from now on we should all proceed from the *lowest common denominator* approach – don't assume the user can do anything that his or her computer can't.

They should really buck up, though, because in fact, human users possess powers that can be of great use in saving programmers unnecessary headaches.

The key skill is being able to take a *general* map and place a pivot point on it for the center where the user *thinks* – but doesn't absolutely *know* – it should go, then using the adroit zooming faculties of the computer to run with that virtual pivot and doggedly follow it on a straight vector – and *have the human be right*, having used instinct to determine what area should be zoomed.

Of course, people aren't infallible either, but clearly their brains work differently from the binary logic of a computer (as Von Neumann pointed out late in his life) – and humans have different *mandates* than a computer. I don't absolutely need to *know* what all of our mandates are to sense it – a human tends to arch over a subject whereas a computer knows nothing *of* a subject except what it's told – and then it only wants to pursue the subject in *other* ways it's told.

This is an ace in our hole, but the Google people, in their prejudice that users should only want to zoom *out* from pre-determined *minutae* (with a pivot zoom) and not easily be able to *set* a faithful pivot point from an overview of the map and stick to it – ignore our human qualities, and make things more difficult.

Once again, Neave's 'Flash Earth' gets it right where Google had gone wrong.

8 - Glide – Over to Assign Data Tags to Masses of Files Quickly

The very least that animators *get* about the world is the value of a second and parts thereof. Animators (most of them, anyway) hate *latency* – that poor excuse for a slow drag through frustration which plagued early digital interfaces. It still does, to an extent. Animators know that the difference between success and fizzle can be *faster* than the blink of an eye – success can be determined by single-frame increments, spanning no more than 1/24th of a second.

Fizzled microseconds are the most obvious thing they hate, but from that grasp of the value of small intervals comes a few other targets. I can throw my hat into the ring here. I hate the cumbersome repetition of obvious tasks, which adds up in aggregate, driving people away from investing any effort. In aggregate is a key concept here – a built-up latency of two 24ths of a second on a second-by-second basis is impossible to ignore, like two clock-ticks out of sync.

Yes, I'm talking about the current 'accepted' method by which Windows 'allows' (for lack of a better word) users to re-name the sluice of small files generated by most modern applications.

In effect, Windows *doesn't*. This has to change. The storm of new, small files – and the places to put them – is a wild modern phenomenon, and whenever something overspills its banks like that, bottlenecks tend to emerge.

In these modern times, one of the biggest bottlenecks (or, in fact, smallest) consists of the way small files – let's take Jpegs, for example – are named by the user in Windows. It's roughly the same way as in 1984. *It has not changed*, not gotten faster, friendlier, more efficient – nothing.

Some programmers (definitely not animators) may write this off to the notion that they're perfect. Well, they're so wrong it hurts. It literally hurts your wrist and fingers after you *squint – aim - right click – select rename – maneuver to the proper section of the renaming string – carefully delete the previous subsection – carefully type something new in without benefit of a tool board of standard terms – check for mistakes – either click on a white spot or press 'enter' which might not always work – search for the new place for the listing – check it went off okay – repeat.*

That was the process for renaming *one* file.

In the old days, when a few dozen Jpegs might push the limits of your hard drive's storage space – you know, the days of 1984 – this was acceptable. If a user, then, were to langorously fuss over the renaming of ten files in a morning, that would take care of most of their workload for the day. Not so anymore, when a trip to the beach might produce 200 new Jpegs for the processing.

As programmers know, a user had more time, in 1984, to think about and bask in the greatness of programmers and their acumen – and the fact that they communed with these bright lights of the computer world simply by virtue of being a user.

The user's thought process, it was assumed, would go like –

squint
Wow! I love this new 17 – inch monitor! And how do they get the type so small? It's almost like reading a book or a newspaper or something! I love this technological age!

aim
This new 'mouse' thing is awesome!

right click
Incredible. What won't they think of next? *Two* mouse buttons! Will wonders never cease?

select rename
Look at the fantastic menu of options! It's like visiting a fancy restaurant!

maneuver to the proper section of the renaming string
Cool! I love the way the 'cursor' is selecting stuff, and how it looks like an arrow!

carefully delete the previous subsection
This is how real 'copy editors' do things! Isn't it amazing I can *go back* and change things, and they erase – ping – just like that, when I hit backspace, or one of those other great 'right click' type commands? Technology truly ushers the common person into the new age!

carefully type something new in without benefit of a tool board of standard terms
Like I said, it's just like I'm bossing people around at a newspaper! Yet it isn't people – but this, my newfound robot slave buddy! I think I'll retype this subsection of my Jpeg name *over and over*, just to bask in the warm glow of technology and its charms!

check for mistakes
Hey, no biggie if I make mistakes here. You know what happens then – I 'ping' it out and 'clickety click' it back in again! It's so much easier than a ribbon typewriter!

either click on a white spot or press 'enter' which might not always work
Cool! At the touch of a mouse button, stuff changes! Who cares if it doesn't always work – I'm the mouse master! I'm controlling a brand new, technical piece of electronic equipment!

search for the new place for the listing
Isn't that amazing that *without my telling it to*, the machine knows to re-juggle everything so as to put it in alphabetical order? I'll bet the stuffy old librarian in the 30 – year old skirt could never do that!

check it went off okay
That's such a cool 'scroll' effect! It's like I'm moving a piece of paper up and down, except I'm just sliding the 'mouse' up and down the side of the screen! I think I'm just gonna fiddle with that for the next half day! The other 3 Jpegs in line for the same treatment can wait! I love technology!

Well, people might have spent a few afternoons that way without minding it. Do it several times in a row, though, and it starts to grate.
Much like your first few bites of a really long candy bar, it carries a kind of thrill – for a few seconds.

Soon enough, the glory of technology wears off and what's left is a teeth – grinding series of tortures, only loved by programmers who like the fact that people scream their name in the process of running the gauntlet, driven to distraction.

Instead of dozens, people now have *tens of thousands* of Jpegs – to say nothing of less manageable small files such as MP3s – and technology's cutting edge makes the user bleed.

Well, in order to patch up the blood flow with some gauze, a new breed of high – tech naming tools is required.

These are, in fact, only high tech in the sense that they haven't been tried before. The technology for them has existed at least since 1990 – or whenever GUI interfaces broke into the mainstream. It would require *no* new investment in hardware, just smart use of the platforms that exist.

Here are some elements that the new approach should contain –

The user should be able to navigate using a minimum of clicks.

a 'catalog tree' should be set up – looking like a series of interconnected tines branching off the main one. This should represent the entire file system overview of the hard drive. This sort of thing has been absent from Windows for some time, and the main reason is probably that the people who might have designed it were *programmers, not visual artists, dammit* – and could only think in numerical terms.

This 'catalog tree' would allow users to *clearly indicate* to the interface just which *batch* of Jpegs (using Jpegs as the example here) they're dealing with. This is a logical approach since often, a user goes about mass renaming projects *when there is a new influx of files* – let's say they just cleared out the memory of the SD card inside their digital camera. Once they *see* the batches in their folders, a clear – cut method should exist to indicate which one they're handling.

The user should see, in smaller form, what they're dealing with.

The programmers of Windows will argue that users already *can* – in the form of the extremely useful 'thumbnail' setting on the views in each folder. Well, maybe – but no effort is made in the design of Windows to unite the function of *viewing* these squares with *naming* them – or doing anything else with them, for that matter, including to be able to view them in a series of tiles that move in motion arcs, and don't skate all over the page like tadpoles on a skillet whenever the user readjusts the aspect ratio of a window.

There's also the slight problem that, in practical terms, the actual displayed size of each thumbnail (which does not change in Windows) is *just* shy of the area which would allow most users to effectively process the content of the Jpeg in their minds, on most monitors.

The common default in programming has been to *keep functions separate* – to have one process for viewing, one for renaming, etc. – this also has to change, and fast. All it requires is to design an interface that acknowledges the underlying *reason* for a user to want to view vast numbers of tiled Jpeg thumbnails – they are often *viewing to rename* – or, at the very least, to try and make sense of the mass of them – which can only be done using some long – term notation.

What we have is something like this:

nailing hammer | prying-up hammer | splitting hammer

The current Jpeg-renaming procedure can be represented by these 3 hammers in a carpenter's toolkit. One is only for pounding nails. Another, for prying things up. The third is only for cracking or splitting objects.

They are so designated, of course, to play up the fact that they are all basically the same hammer.

A 3-hammer holster designed by Microsoft might specify that none of the hammers can be used for any but their primary use. No prying things with the nailing hammer, et cetera.

This is about as smart, unfortunately, as the current Jpeg-handling procedures – where DIFFERENT programs and processes (all valid in and of themselves) are to be used for the closely-related functions of a) viewing thumbnails, b) viewing the Jpegs full-size, c) renaming them and d) allocating them to different folder locations.

Forcing a woodworker to *switch hammers* everytime the need arises to do something different with a nail would add huge amounts of time, as well as sweat, to the woodworking process. It might make for work that's twice as long. Fortunately, since hammers are physical objects, referenced on an analog plane, their users can swiftly overcome artificial divisions of use of a single object. All a woodworker needs to do in real life to switch uses of their hammer is to flip it around. No such solution is available to users in the cut-and-dried digital world of software interfaces, where the programmer wields absolute power, and tools can be as simple – or as complicated – as any programmer feels like making them.

In short, users want to display, analyze, and rename thumbnails – fast. The users can almost always do that. The interfaces, though, need to catch up.

There are several ways in which a new interface has to go over and above the one offered by Windows – first, it needs to make the tiled thumbnail display *less slippery* – in that they need to have designated *places* within the visual field that thumbnails rarely 'slide around' from – *unlike* the Windows approach, where the thumbnails are treated, essentially, as glorified data.

With the 'glorified data' approach, the placement of thumbnails in the interface is an indifferent matter. Thumbnails indelicately 'blink' in and out of rows, depending on the motion of the user's mouse – controlled cursor towards the bottom right of whatever window they're in. Simply by clicking and dragging to increase or decrease the size of the frame's aspect ratio, a user inadvertently 'juggles' the thumbnails, shifting their place with relation to each other.

What this results in visually is the inability of the user to rely on something being where they left it from second to second. In terms of animation, Windows presents this re – juggling without any 'inbetweens', with no follow – through, weighted transition between placements, or arcs of motion. While the 'resize window' function has some of the above, the positioning of the thumbnails has none – once again, a sorry division of function – making for a skittish looking interface with a kind of 'Atari 2600' feel, the icons 'blipping' from position to position with no warning and only a slight indication of direction. The user's given no *visual* 'tab to grab' – no cue telling them, in a reliable way, just where any of these hopped-up thumbnails is bound.

The newer approach should include a visual field that's as safe and reliable for the thumbnails as a page in a book – that is, a field where the thumbnails are tiled in an appealing fashion, *consistently*, without fear of slippage (words rarely get 'juggled' around book pages), providing areas nearby for pertinent information (such as each Jpeg's title and date), and ample maneuver room for the user.

The thumbnails should clearly indicate what they are small pictures of.

Once again, programmers will argue that their primary purpose is to indicate that. However, the simple fact is that most times when the thumbnail view setting is activated in Windows, the slippery, ill – placed little tiles are just *too small* to convey the proper amount of visual data to allow the user to make an informed decision about how to name or catalog them.

This is an area where graphic design also plays a part. Programmers might balk at the concept of a thumbnail re – design, since, after all, they've been held at their current dimensions since the last time any standards for them were established. The good news is that you *don't have to* re – size the thumbnails from the point of view of resolution – just *make their surface area larger*, and then the user will be able to see more *in* them, regardless of resolution. That's just the way it works.

It's true that a full-bodied approach to the sizing of the thumbnail within the window will require a consideration of current settings governing screen and pixel dimension. This might give software designers pause to think. Fortunately for programmers, however, the two interfaces – that of Jpeg thumbnail generation *and* that of screen size and aspect ratio regulation *both* exist as functions of the Windows OS – and, so, the data set of one will easily be available as reference for the other.

The approach that treats thumbnails as glorified data really doesn't concern itself with *actual sizes* of the visual image on the page. According to the programmers who established the Windows method of displaying thumbnails, the function of shuffling them around to compensate for changing window size and the function of making them displayable should exist independently of merely getting them there in the first place. They leave the visual approach to someone in a different department.

Most importantly – the user should be able to assign data without clicking.

Most programmers will react with 'what?' It's true, though – the shining darling of the mouse programmer's trade, the 'click', is a major – league pain in the rear when it *must* be done to accomplish simple tasks. Chief among these tasks, when dealing with *huge* amounts of new and small files, such as Jpegs, is the task of assigning data such as names and dates to them.

A simple glide over the visual placement of these thumbnails should always result in the entry of data, or the possibility of it.

Some alternate tools should be available, such as *extremely simple* keyboard commands – say, 'enter' for skipping to the next frame of reference, or integration of the arrow keys – but nothing should be allowed to complicate the eminently simple 'glide over' approach, reducing the mouse to a *selection* tool rather than a data entry tool. The misconception of all programmers who integrate the mouse into a program is that the user is enraptured with the click function.

The click function is not necessary for naming, except in the merest sense that it will sometimes be useful in specifying 'this, not that' or selecting a pre – arranged bit of data to incorporate into the overall body of a string of data. For the simple act of selecting items, though – clicking is overused.

This phenomenon is a part of animation as well – in that *small actions* have a way of stacking up when used, not even following one another immediately, but *interspersed* at regular enough intervals between other small actions.

In animation, you can have somebody scratch part of their arm *during* a walk cycle. That's sort of like the occasional click. However, if you make the 'scratch' action an integrated *part* of the walk cycle, it becomes like a nervous tic – hard to ignore and off – putting. This is the relationship the user has with clicking. If it's once in a while, fine. If it's 30 times a minute, most users will abandon whatever project it is just to get away from the frenzied click motion. I know I've declined to amend many a generically – assigned Jpeg name for just this reason.

The assignment of data should be very simple for the user.

The simplest way to make data entry simple is to take advantage of two natural strengths of a modern computer system. These are both the ability to quickly copy and paste, and the ability to concisely display a series of clickable options on the *side* of the visual field of reference that trigger placements of data *on* the visual field of reference.

Basically, what it boils down to is to display a visual toolkit of data entry tricks for the user – a 'cliché belt' – which simplifies the grammar of data strings. The user should be encouraged to enter their own preference for the naming of these chunks of data.

This sidesteps all issues of re – tooling the Windows clipboard. It might be nice if it could incorporate more than one data string at a time – but with current varieties of visual interface, it doesn't even need to. All the user would need is a simple 'click – board', or tool belt, of data snippets:

Cottage '04
Our Trip to Beach
Al and Sue
Me with Hat
Skippy's Haircut '07

and so on – these would be somewhere off to the side, clickable, determined by the user – and would prove valuable tools in assigning titles and other data to Jpegs and other files.

The theory here is that the user will view to re – name when there is a *new influx of files* – but also that the new influx will consist of variations on similar themes, which only a few tags will typify.

The overall purpose of such tools would be to *allow the user to use what they've got*, through smooth and accessible interfaces. If more rationale is needed, however, it could be that a user who's well catered to by the receiving tool will then demand more of the giving tool – that which they buy.

The overall objective of this type of system would be to reduce the squint – aim - right click – select rename – maneuver to the proper section of the renaming string – carefully delete the previous subsection – carefully type something new in without benefit of a tool board of standard terms – check for mistakes – either click on a white spot or press 'enter' which might not always work – search for the new place for the listing – check it went off okay – repeat process to something more like

hover – select – enter data – advance…

and so on.
Another way to describe this method – modern users are faced with huge dirt hills of data. Currently they have tablespoons, and the aim of this is to give them steamshovels.

Do you know where your cursor is?

Surprisingly, the most central of GUI input devices – and the sole ambassador of people in the realm of the computer – that is, the cursor on-screen – is almost always disregarded or forgotten by programmers. The cursor represents the will of the user *and* indicates the area where the user's attention is currently directed (the user will almost always be *looking at* the cursor when pointing it somewhere) – and, by extension, provides any alert programmer with cues as to the user's long-term intentions.

Programmers owe it to themselves to *always* be aware of the cursor's placement, and use it as the primary interface point – as well as to expand its privileges. Always be aware of the cursor's placement. Using this, it can be the largest boon to both programmer and user.

By using the cursor as the focal point it *already* is, the programmer can facilitate great things: Cursors could be used to indicate where a fisheye effect should go – in fact, could be used in conjunction with one key (ensuring, of course, that it's an easily-pressed key, and one that the programmer sees to it will *be* pressed when pressed, not slip into the coyness of refusal, as indicated in the top table) to allow the user to *insta-zoom* or insta-magnify a given radius of the interface on a whim, thereby eliminating cumbersome text wrangling and squinting.

9 – A Few Notes on Maps

A few notes on maps:
We see the idea of accuracy in geographical maps as sacred, mostly because of the fact that we're just coming to understand how accurate they *are*, and don't want someone messing with it.

Just fine – although it sometimes prevents us from understanding what more maps can offer, and this comes to us in the form of the *political map* – or any map which dwells on organization of human detail rather than precision in physical landscape.

Be it resolved – a political map of any kind has a use that transcends the purely physical, and so should be viewed differently and explored as a new device. It tells the reader *what's going on* over the span of the locale it deals with.

Both maps are necessary. The physical is the kind we've just begun to really come to know, through web applications. We like embracing it. But we ignore the political type at our peril. We run the risk of convincing ourselves that whatever doesn't 'amend' the raw physical attributes of a place isn't to be dealt with. That eliminates consideration of most of our daily lives – and under-credits people as masters of their own world. It ignores the past century and a half. You could say that these types of maps are the hidden guidebook to human behavior and society.

The challenge that map makers are faced with, in the near future, is to evolve custom – made varieties of political type maps that are useable, pleasing to view, and non-offensive – that is, which make themselves useable without stepping on any significant toes.

What that alludes to is the formidable power maps seem to have carried in society from early on. There have been many occasions where the ruling elite have struggled to ban them. To do so is to impose a 1984 of the spirit by censoring basic human desires. We shouldn't concern ourselves with the petty paranoia behind such things, however. The map, in useable form, provides a boon to explorers, historians, and the lay person alike. Its increasing prominence as metaphor for personal advancement is no mistake.

This also indicates the value of the political map with changing (animated) vector attributes scaled along a timeline. It will bring into relief a kind of history which could previously only be told – and clumsily, at that – in multiple hundred page tomes, and establish a view of the future and past previously reserved for the planner elite who had access to specialized data, as well as unlimited time to pore over it. It's about time for a democratization of such elements.

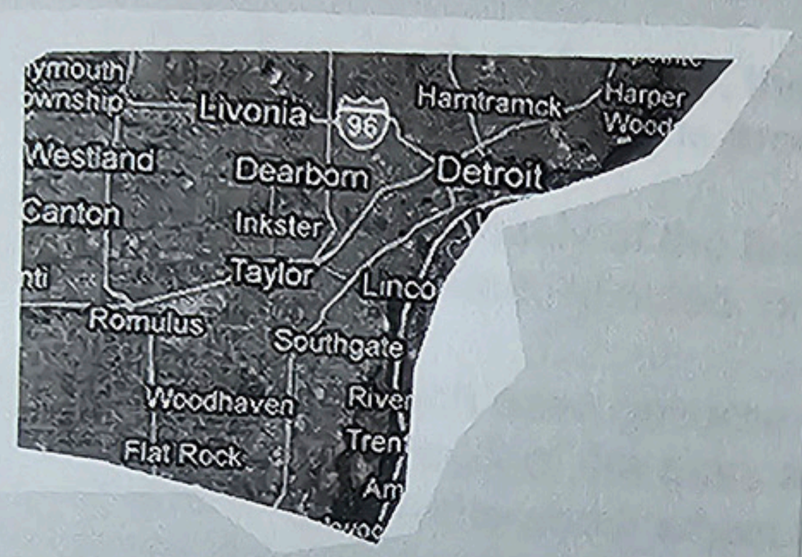
A good political map can be the ultimate guide to man – made attributes of a landscape. By that I mean all the places and spaces which are vital to daily commerce. The day a citizen unlocks a steady and easily perused simulacrum of his or her favorite street or shopping district, and it

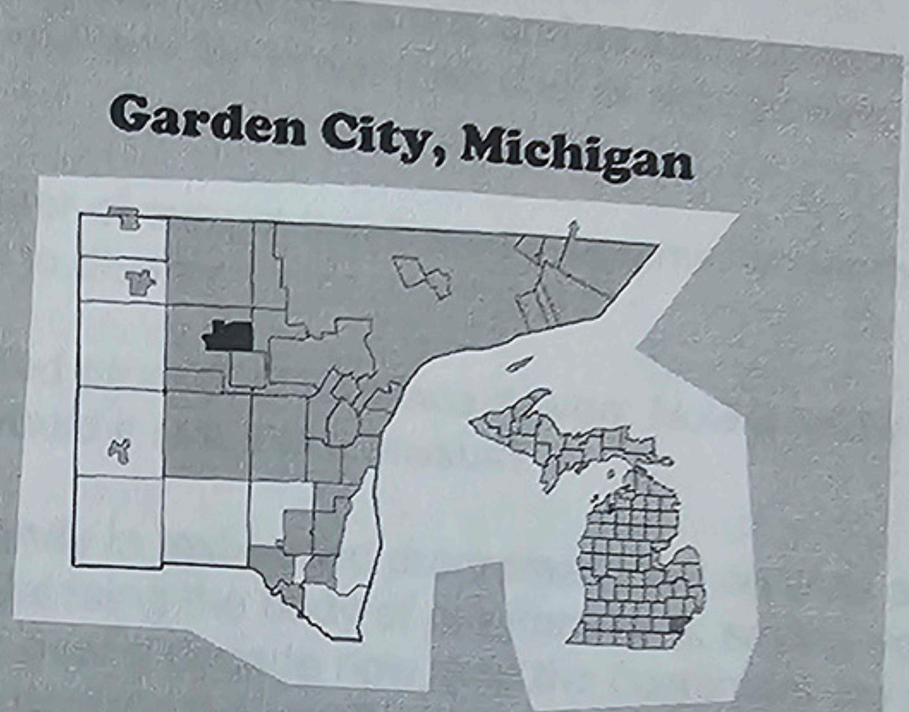
concisely displays the totality of a user - defined category of data (for instance, locales of tennis courts or Chinese restaurants) is the day freedom is tasted.

These maps, well – laid out, could easily replace every other form of categorization currently in use – tourist maps, phone directories, any guide to the listings of services a person could need throughout the day in a region. They will do this, too, without much amendment to existing media or technology.

example:

Which map gives the reader more information?





On the left, you have an abundance of *detail* and nominal and physical accuracy. On the right, you have clear lines and labels *guiding* the reader to elements of a data set. Detail or clarity? I'll let you decide.

10 – A Little About Search Engines

A little about search engines.
While the nature of romantic portraits of the 'net (world wide) can deny it, *locality* is coming to searches – and sooner is better for this.

Google seems to have reasoned, of late, that a) people spend most of their money locally, *even if it's on foreign products*, and b) the main reason someone even bothers with a search engine is to connect them to a locality – whether it's a search for that perfect park, your hockey team schedule, what's playing at the theatre across the way, the relationship status of that girl from the drugstore, or whatever else.

With this realization, they have given priority status to any result that contains the terms of search *plus* the user's hometown or locale – as calculated by IP address, server details, or things like previous search activity.

There's a novelty factor to the cladding of buildings in Norway, or what Calgarians think of their nightspots. Most people, though, never care about any of that. We also want to believe that science leads us to ever greener pathways, and that, following that logic, we're all on the way to being *global citizens* – whatever that really means. Nevertheless, we remain fiercely location – based animals.

This isn't really *propelled* by the high potential of maps and location – based searches in connecting a user with their preferred piece of data – the two evolved side by side. The fact remains that, right now, computer interface designers have the *ingredients* for hundreds of barrels of TNT directly under them. Few take the time to synthesize it, though.

A good test for the effectiveness of any interface: *Is it a boon*? Does it contribute something significant to my life or open doors that would have remained closed?

The answer to this, where Earth – based locales in search criteria are concerned, can only be yes.

11 – How Would it Be With a Car?

Here's an area where interface designers actually work without a net, and as such, is worth studying. Automobile design allows little room for error, and less for 'horsing around' or acting clever to the exclusion of practicality.

As such, it's a breath of fresh air for, say, the user of current text-based programs, where the programmer's cleverness always seems prone to display.

A question to be asked relentlessly of the finished product of interface design: *How would it work with a car*? Would it be tolerated, rejected, or would it result in a lawsuit?

Take the ribbons up top which have become trendy in text-based programs: Their controls are probably an over-reactive result of the boss addressing the body of programmers, saying 'you guys have been content to ignore the scroll wheel for over a decade now, and the customers are starting to burn our offices down in protest. I insist you integrate the scroll wheel from now on!'

Well, programmers, with egos to defend, half-heartedly took the suggestion seriously. What they came up with was a ribbon which definitely acknowledges the presence of the wheel – and then proceeds to use it for all kinds of overkill, as if to retort, 'dammit, look what you made us do." They seem to want to rub the boss's face in their bossy nature.

The most evident way it goes over the edge is in that, with the cursor placed anywhere on the ribbon, *you can't actually avoid using the scroll wheel*. If the scroll wheel happens to be in motion – even a bit – a new 'leaflet' of the ribbon's functionality is selected. This creates a kind of fan-next-to-paper effect, where one element of the desktop (the currently-viewable 'leaflet' of the current ribbon) is constantly changing in barely predictable ways. Great.

So: How would it be with a car?

Well, let's say, for instance, that the dashboard had a 'ribbon' feature on it: What if the dashboard of your new car were nothing more than an LED digital display? What if, further, the functions on it – stereo controls, speedometer, odometer, fuel level, temperature – were broken up into 'virtual pages' which the driver could virtually flip?

Already there are a few problems that crop up: What happens if the driver wants to see every vital function at *once* – you know – like on regular dashboards?

Well, minor adjustment for tomorrow's driver: The programmers of this new LED interface have given the driver a hands-free method of 'flipping the page' to access whichever vital driving-related function they desire at the moment: The dashboard is wind-sensitive, so that the act of lightly blowing with the driver's mouth on one or the other side of the wind-sensitive LED board will flip to the next virtual

page, revealing the desired function. Of course, the driver will take a little while to get used to it, but that can be done.

Well, the obvious end result – practically so obvious that you wouldn't even need to be a driver yourself to anticipate it – would be this: The first time a breeze blows in from the outside window, your dashboard gets screwed.

The problems with a given approach seem much more obvious when you project the possible future disadvantages: With a computer, you might simply be stifling the creativity of a budding writer, or making a designer or casual user's life a living hell – or a series of petty annoyances. While some would hold that to be enough reason *not* to do a given thing, the refusal most of us reserve to project the consequences of anything we do much past the nose on our face means that the feature, once designed, will probably stay for a while. It's the tired-teacher effect: The kids have just produced twenty slick, sloppy, less-than-pretty fingerpaintings on construction paper, and left the class. Now these paintings are mine to do with what I please. Should I haul them all to the incinerator? No, that would take more effort than just hanging them up here. On the classroom walls they go, see if I care. Let the tots deal with the aesthetic effects of their work hanging over their play area until the summer. If an asinine design feature makes its way onto the dashboard of a car, though, the consequences are swift and direct: Accidents happen and people die. Yeah. Maybe worth changing it for the possible-death arena.

How about the flighty, skate-y YouTube volume interface? The one that disappears into invisibility with a slight arc of motion when not being used, resulting in confusion to the untrained observer, and fumbling for position whenever the feature is used?

With YouTube, the immediate predictable result is that certain teens will tear their hair out trying to gain easy access to illicitly-uploaded media. Ho-hum.
With a car, however – well, consider the analog: An automatic, foldable steering wheel. It cleverly hides itself inside the surface of the dashboard, retreating, like a turtle neck, into a cleverly-designed robotic door, when the car's in park or when the driver takes their hands off the wheel while driving. This gives the driver a cleaner-looking dashboard effect, a certain percentage of the time, as if the wheel had never been installed – looks better in photos. It's an effect close to the one attempted by the newest YouTube interface designers, who surmised that users wanted a volume control, but didn't really want to be able to find it at a glance – it'd be uncool. Heat-detection panels would sense when the driver's hand approaches the robot doors, (when they're fumbling for the wheel again), and oh-so-cleverly un-turtle it – extending it like a basketball player's high-five to meet the insistent flesh of the driver. This might happen twenty or more times during the course of driving, brightly illuminating the triumph of man over technology, as well as the car designer's innate cleverness. Unfortunately, it will result in accidents where people die. Yeah. Maybe worth changing it for the possible-death arena.

And coyness of keys? Let's say, 20% of the time at random, the right-flasher *doesn't work* when you're signaling to turn.
There's no way of predicting whether it will work or not ahead of time. It's just that, sometimes, the car appears *deaf* when you tell it to signal a right turn. Let's also say that the remedy for this is to (virtually) 'shout' at it, admonishing it to smarten up and signal, by hitting the appropriate lever (on average) 3 more times in rapid succession.

No different, really, from the lax attitude most programmers take towards keyboard commands. Many of them seem *truly indifferent* as to whether your command input is registered through the keyboard – even going so far as to build in 'selective hearing' mechanisms into the program, mimicking someone whose hearing has gone bad, which reliably require 3 to 4 repetitions of the same keystroke in order to register a single one. Mouse clicks aren't in any way immune from this,

either. There's one program whose procedure for registering one click on a given field is as follows: a) click. b) frustration. c) click-click-click. d) (if you're lucky) a registered click.

In a car, this *would* successfully allow cars to become more like people – thus contributing to the robot revolution, bridging the gap further between humans and cyborgs, and imbuing cars with characteristics typical of 'Rosie the Robot' off of the Jetsons – which gained some popularity with the TV-watching public – insouciance, sarcasm and the like. Unfortunately, it will result in accidents where people die. Yeah. Maybe worth changing it for the possible-death arena. And also for all interfaces, while we're at it.

Concessions

All this comes from a sideliner, a non-programmer. Well, not entirely. I've done Applesoft Basic – way back when – and, while it did give me an entrance to the scene, it prepared me very little for today's languages and meta-languages.

For all I know, there are a host of internal conflicts within an organization such as Microsoft which make inter-departmental teamwork hard. I've also been slow to sell my skills as a programmer, such as they are. The simple fact is, programming – in the modern, graphical interface related sense – is really just in the process of getting off the ground, and anyone making inroads into it, who is reaping a paycheque from it, is in a different league from me, and can offer lessons from experience.

As for the more skillful deployment of graphical user interfaces, it must be hard. I will admit that I have downloaded and installed several trial versions of several software 'solutions' to the blandness of the Windows desktop. None have worked. While they gained significant points for flashiness, they lacked a certain kind of solidity.

One, in particular, stands out, if only for how promising a concept it was. It was called BumpTop, and was a 3D approach to the desktop – something I would argue is overdue from Microsoft – but not all that elaborate, simply an attempt at re-crafting the graphic look of the standard desktop in a 3D context, by simulating a kind of 'room' appearance (more like an open-topped desk or something) where the formerly flat desktop icons could take on the form of 3D objects.

It was a great idea, and to make it greater, there were action based commands you could execute with the 'objects', essentially duplicating existing windows commands for shuffling of files, but including a simulated physics which appeared to give weight to the desk objects, and the ability to have them push and affect one another.

The icing on the cake was something I have wanted for a while: The ability to re-size the objects at will, making some appear to have greater impact than others.

It would all have meshed together into an unbeatable package, except for the bugs. The bugs did it in. More than once it crashed the system – something like my first few installation attempts at Google Earth. I was thrilled with some parts of it, and the whole idea of someone in a small company out there having the guts to challenge Microsoft at their own game seemed overdue. But what the programmers had written out of their equation was the fact that, over the years, Microsoft has built up a desktop infrastructure.

The Windows OS is ingrained into the framework of the PC platform so intimately that anything trying to piggyback on it, such as this new desktop program, needs gross amounts of pre-testing and inside

involvement from Microsoft itself, to prevent it facing a hard uphill climb. It would be nice to believe that someone with a bit of programming skill and a few snappy ideas can challenge the big dogs – but, as time has told, it wasn't the case.

I will also admit that the Windows OS – so much a part of this platform, from the beginning, that it has become a standard sport to take potshots at it – does have an underlying solidity to it. While people can still justifiably come up with this and that complaint or suggestion, the admission has to be made that it's probably a more stable interface than most out there. The earlier versions seem even more stable than later (with the exception of 7, which seems to have de-escalated some of the snappiness layered on to this one, Vista, for the sake of re-capturing some of the wholesomeness people saw in the earlier version, XP). It definitely holds up well against the OS which conceivably inspired it, the successive versions of the one installed in Macs and Apple products.

So, I guess many criticisms I can make can be summed up by simply saying that these interfaces aren't pretty. That, apparently, was never the intention of Microsoft. The people at Apple do seem to put a higher value on aesthetic success, and so have prioritized certain flourishes aimed at prettying-up their desktop and other interfaces. I have to applaud this. I'm sorry, too, that in my most recent years of PC usage, I have missed the chance to become more familiar with these tools, so squarely aimed at those familiar with graphic design.

Let's just say this, overall: There is much growing room in current interfaces, and much of it relates to the way graphics are employed to communicate clearly and directly with users to achieve goals. I'm not sure I see anyone boldly leading the way forward into a graphics – dominated future – but then, the future also needs a lot of infrastructure, and needs to be safe. It's up to everyone involved to strike the right balance, looking to the edge as well as the spinning center of the kaleidoscope for flashes of the way forward.